Computing Stationary Distribution Locally

by

Christina Lee

B.S. in Computer Science, California Institute of Technology, 2011

Submitted to the Department of Electrical Engineering and Computer Science
in partial fulfillment of the requirements for the degree of

Master of Science
in Electrical Engineering and Computer Science
at the Massachusetts Institute of Technology

June 2013

Signature of Author: _____

Christina Lee
Department of Electrical Engineering and Computer Science
May 22, 2013

Certified by: _____

Asuman Ozdaglar
Professor of Electrical Engineering and Computer Science
Thesis Supervisor

Certified by: _____

Devavrat Shah
Professor of Electrical Engineering and Computer Science
Thesis Supervisor

Accepted by: _____

Leslie A. Kolodziejski
Professor of Electrical Engineering and Computer Science
Chair, Committee for Graduate Students

Computing Stationary Distribution Locally

by Christina Lee

Submitted to the Department of Electrical Engineering and Computer Science
in partial fulfillment of the requirements for the degree of
Master of Science

Abstract

Computing stationary probabilities of states in a large countable state space Markov Chain (MC) has become central to many modern scientific disciplines, whether in statistical inference problems, or in network analyses. Standard methods involve large matrix multiplications as in power iterations, or long simulations of random walks to sample states from the stationary distribution, as in Markov Chain Monte Carlo (MCMC). However, these approaches lack clear guarantees for convergence rates in the general setting. When the state space is prohibitively large, even algorithms that scale linearly in the size of the state space and require computation on behalf of every node in the state space are too expensive.

In this thesis, we set out to address this outstanding challenge of computing the stationary probability of a given state in a Markov chain locally, efficiently, and with provable performance guarantees. We provide a novel algorithm, that answers whether a given state has stationary probability smaller or larger than a given value $\Delta \in (0, 1)$. Our algorithm accesses only a local neighborhood of the given state of interest, with respect to the graph induced between states of the Markov chain through its transitions. The algorithm can be viewed as a *truncated Monte Carlo method.* We provide correctness and convergence rate guarantees for this method that highlight the dependence on the truncation threshold and the mixing properties of the graph. Simulation results complementing our theoretical guarantees suggest that this method is effective when our interest is in finding states with high stationary probability.

Thesis Supervisor: Asuman Ozdaglar
Title: Professor of Electrical Engineering and Computer Science

Thesis Supervisor: Devavrat Shah
Title: Professor of Electrical Engineering and Computer Science

Acknowledgments

This thesis owes its existence to Professor Devavrat Shah and Professor Asuman Ozdaglar. It has been an incredible joy to work with them both, and I could tell stories upon stories of the encouragement and inspiration they have been to me. They have been incredibly patient with me and always put my personal growth as a researcher first. I have learned so much from them about how to approach the process of learning and research, and how to face those moments when progress seems so slow. I remember countless meetings in which I walked in frustrated, yet walked out encouraged and excited again. I appreciate Devavrat's inspirational analogies, reminding me that grad school is a marathon, not a sprint. I'd like to thank Asu for her warm support and patience in teaching me how to communicate clearly and effectively.

I am also incredibly grateful to the LIDS community. Truly the spirit of the LIDS community matches the bright colors of the LIDS lounge, always able to put a smile on my face. A special thanks goes to my great officemates Ammar, Iman, Yuan, Katie, and Luis for sharing both in my excited moments as well as my frustrated moments. I appreciate their encouragement and reminders to take a break and relax when I am too stuck. I am grateful for Ermin, Annie, Kimon, Elie, Ozan, and Jenny as well for bouncing ideas with me and encouraging me along in my research as well. I am in debt to the amazing community of LIDS girls: Shen Shen, Henghui, Mitra, and Noele, who have colored my days and made my time here full of laughter.

A big thanks to the Graduate Christian Fellowship at MIT and the Park Street Church International Fellowship, which have become my family in Boston. As always, I am forever grateful to my parents unending support and my siblings encourgement. Finally, the biggest thanks goes to my greatest supporter: to God who reminds me every day that I am his beloved child. His love frees me from unhealthy pressures and fills me with joy and strength each day.

Contents

List of Figures

Chapter 1

Introduction

1.1 Motivation

The computation of the stationary distribution of a Markov chain (MC) with a very large state space (large finite, or countably infinite) has become central to many *statistical inference problems.* The Markov Chain Monte Carlo (MCMC) coupled with Metropolis Hasting's rule or Gibbs sampling is commonly used in such computations. The ability to tractably simulate MCs along with the generic applicability has made the MCMC a method of choice and arguably the top algorithm of the twentieth century.[1] However, MCMC (and its variations) suffer from a few limitations especially for MCs with very large state space. The MCMC methods involve sampling states from a long random walk over the entire state space [18, 25]. The random walks need to be "long enough" to produce reasonable approximations for the stationary distribution, yet it is difficult to analyze and establish theoretical guarantees for the convergence rates. Furthermore, a large enough number of samples must be used in order to ensure that the distribution has been fully sampled from.

Stationary distributions of Markov chains are also central to *network analysis.* Networks have become ubiquitous representations for capturing interactions and relationships between entities across many disciplines, including social interactions between

[1] The Top Ten Algorithms of the Century.
http://orion.math.iastate.edu/burkardt/misc/algorithms_dongarra.html

individuals, interdependence between financial institutions, hyper-link structure between web-pages, or correlations between distinct events. Many decision problems over networks rely on information about the importance of different nodes as quantified by network centrality measures. Network centrality measures are functions assigning "importance" values to each node in the network. The stationary distribution of specific random walks on these underlying networks are used as network centrality measures in many settings. A few examples include PageRank: which is commonly used in Internet search algorithms [29], the Bonacich centrality and eigencentrality measures: encountered in the analysis of social networks [9, 10, 28], rumor centrality: utilized for finding influential individuals in social media like Twitter [32], and rank centrality: used to find a ranking over items within a network of pairwise comparisons [27].

The power-iteration method is currently the method commonly used for computing stationary disributions of random walks over networks, such as PageRank. The method involves iterative multiplication of the transition probability matrix of the random walk [16]. Again, this method suffers from similar limitations for large networks: there is no clearly defined stopping condition or convergence analysis for general settings, and it requires computation from every node in the network. While this can be implemented in a distributed manner using a message passing algorithm (which involves computations that use information from neighbors of a node), convergence rates are often hard to establish.

The massive size of the state space of these Markov chains is the primary reason for development of super-computation capabilities – be it nuclear physics [24, Chapter 8], Google's computation of PageRank [29] or Stochastic simulation at-large [3]. In the presence of Markov chains with massive state space, an ideal algorithm would be the one that overcomes both of these limitations. In this thesis, motivated by this goal, *we present a local algorithm to approximate the stationary probability of a*

node exploiting only local information in a subnetwork around that node. By local, we mean two things. First, the algorithm must be easily distributed. The algorithm must only access the network through neighbor queries, and computation at a node can only use local information passed through his neighbors. Second, the algorithm must be "local" to the nodes of interest. For example, if we are only interested in computing the stationary probability of a single node, then all computation involved in the algorithm must be restricted to this node, where computation refers to any arithmetic operations or operations involving memory. There are many benefits for having such a *local* algorithm.

First, in many settings such as ranking, we are not actually interested in computing the stationary probabilities of all nodes. In web search or movie and restaurant recommendation settings, only the scoring among top nodes are relevant, since users often only view the first page of results. Rather than computing the entire stationary distribution, we may only need to compute the stationary probabilities of a smaller specific subset of nodes. Thus we desire algorithm that scales with the size of this subset rather than with the full size of the network.

Second, there may be physical constraints for accessing or storing the information that prohibit global computation. For example, the information may be stored in a distributed manner either due to memory limitations or the nature of the data. In citation networks, the information is inherently not centralized (i.e. not owned by a single central agent). It may be easy to query a single paper or a single author to view the neighborhood of that node, however performing any global computation would require first crawling through searching node by node to rebuild the network. Many search engines maintain indexes containing the entire webgraph, yet due to the sheer size of the web, the information must be distributed among clusters of computers. Some estimate that the size of the web is growing at a rate of 21% new webpages every year.[2]

[2]Size of Internet and Its Growth Rate.

This causes increasing difficulty for methods such as power iteration, which require computation at every node in the network.

Third, there may be privacy constraints that prevent an external agent from accessing the global network. For example, in many online social networks, privacy settings only allow a member to access his local network within a small distance (e.g. friends of friends). There exist APIs that allow external companies to interface with Facebook's social network data. However, every user must specifically grant the company access to his or her local network. Companies offer users the ability to use new applications or play new games in exchange for the access rights to their social network. Companies design their applications and games to have a social aspect so that after a user adopts a new appliation, he will have incentives for his friends to also join this application. Thus, many of these applications and games are adopted through viral behavior, by friends recommending them to their friends. Thus, a company will often gain access rights to clusters or local snapshots of the network. A local algorithm that only requires information from a neighborhood around a specific node can still provide estimates for global properties such as stationary distribution, using the limited information from these clusters.

■ 1.2 Related Literature

In this section, we provide a brief overview of the standard methods used for computing stationary distributions. Computation of the stationary distribution of a MC is widely applicable with MCMC being utilized across a variety of domains. Therefore, the related prior work is very large. We have chosen few samples from the literature that we think are the most relevant.

Monte Carlo Markov Chain. MCMC was originally proposed in [25], and a tractable

http://levelx.me/technology/size-of-internet-and-its-growth-rate/

way to design a random walk for a target stationary distribution was proposed by Hastings [18]. Given a distribution $\pi(x)$, the method designs a Markov chain such that the stationary distribution of the Markov chain is equal to the target distribution. Without formally using the full transition matrix of the designed Markov chain, Monte Carlo sampling techniques can be used to estimate this distribution by sampling random walks via the transition probabilities at each node.

Each transition step in the Markov chain is computed by first sampling a transition from an "easy" irreducible Markov chain. Assume the state space is represented by an n-length 0-1 vector in $\{0,1\}^n$. A commonly used "easy" Markov chain is such that from a node x, a transition to x' is computed by randomly selecting a coordinate $i \in 1, 2, \ldots n$ and flipping the value of x_i at that coordinate. Then this transition is accepted or rejected with probability $\min\left(\frac{\pi_{x'}}{\pi_x}, 1\right)$. The stationary distribution of this Markov chain is equal to π. After the length of this random walk exceeds the mixing time, then the distribution over the current state of the random walk will be a close approximation for the stationary distribution. Thus the observed current state of the random walk is used as an approximate sample from π. This process is repeated many times to collect independent samples from π_i.

Overview articles by Diaconis and Saloff-Coste [13] and Diaconis [12] provide a summary of the major development from probability theory perspective. The majority of work following the initial introduction of the algorithm involves trying to gain understanding and theoretical guarantees for the convergence rates of this random walk. Developments in analyzing mixing times of Markov chains are summarized in books by Aldous and Fill [1] and Levin, Peres and Wilmer [23]. The majority of results are limited to reversible Markov chains, which are equivalent to random walks on weighted undirected graphs. Typical techniques analyzing convergence rates involve spectral analysis or coupling arguments. Graph properties such as conductance provide ways to

characterize the spectrum of the graph. In general, this is a hard open problem, and there are few practical problems that have precise convergence rate bounds, especially when the Markov chain may not be reversible.

Power Iteration. The power-iteration method (cf. [16, 22, 33]) is an equally old and well-established method for computing leading eigenvectors of matrices. Given a matrix A and a seed vector x_0, recursively compute iterates $x_{t+1} = \frac{Ax_t}{\|Ax_t\|}$. If matrix A has a single unique eigenvalue that is strictly greater in magnitude than other eigenvalues, and if x_0 is not orthogonal to the eigenvector associated with the dominant eigenvalue, then a subsequence of x_t converges to the eigenvector associated with the dominant eigenvalue. This is the basic method used by Google to compute PageRank. Recursive multiplications involving large matrices can become expensive very fast as the matrix grows. When the matrix is sparse, computation can be saved by implementing it through 'message-passing' techniques; however it still requires computation to occur at every node in the state space. The convergence rate is governed by the spectral gap, or the ratio between the two largest eigenvalues. Techniques used for analyzing mixing times for MCMC methods as discussed above are also useful for understanding the convergence of power iteration. For example, for reversible Markov chains, the conductance of the graph is directly related to the spectral gap. For large MCs, the mixing properties may scale poorly with the size, making it difficult to obtain good estimates in a reasonable amount of time.

In the setting of computing PageRank, there have been efforts to modify the algorithm to execute power iteration over subsets of the graph and combining the results to obtain global estimates. Kamvar et. al. observed that there may be obvious ways to partition the web graph (i.e. by domain names) such that power iteration can be used to estimate the local pagerank within these partitions [21]. They estimate the relative weights of these partitions, and combine the local pageranks within each

partition according to the weights to obtain a initial starting vector for the standard PageRank computation. Using this starting vector may improve the convergence rate of power iteration. Chen et. al. propose a method for estimating the PageRank of a subset of nodes given a local neighborhood [11]. Their method uses heuristics such as weighted in-degree as estimates for the PageRank values of nodes on the boundary of the given neighborhood. After fixing the boundary estimates, standard power iteration is used to obtain estimates for nodes within the local neighborhood. The error in this method depends on how close the true pagerank of nodes on the boundary correspond to the heuristic guesses such as weighted in-degree.

Computing PageRank Locally. There has been much recent effort to develop local algorithms for computing properties of graphs [8], including PageRank. Given a directed network of n nodes with an $n \times n$ adjacency matrix A (i.e., $A_{ij} = 1$ if $(i,j) \in E$ and 0 otherwise), the PageRank vector π is given by the stationary distribution of a Markov chain over n states, whose transition matrix P is given by

$$P = (1-\beta)D^{-1}A + \beta \mathbf{1} \cdot r^T. \tag{1.1}$$

Here D is a diagonal matrix whose diagonal entries are the out degrees of the nodes, $\beta \in (0,1)$ is a fixed scalar, and r is a fixed probability vector over the n nodes.[3] Hence, in each step the random walk with probability $(1-\beta)$ chooses one of the neighbors of the current node equally likely, and with probability β chooses any of the nodes in the network according to r. Thus, the PageRank vector π satisfies

$$\pi^T = \pi^T P = (1-\beta)\pi^T D^{-1}A + \beta r^T \tag{1.2}$$

where $\pi^T \cdot \mathbf{1} = 1$. This definition of PageRank is also known as personalized PageRank,

[3] $\mathbf{1}$ denotes the all ones vector.

because r can be tailored to the personal preferences of a particular web surfer. When $r = \frac{1}{n} \cdot \mathbf{1}$, then π equals the standard global PageRank vector. If $r = e_i$, then π describes the personalized PageRank that jumps back to node i with probability β in every step.

Computationally, the design of local algorithms for personalized PageRank has been of interest since its discovery. However, only recently has rigorous analytic progress been made to establish performance guarantees for both known and novel algorithms. These results crucially rely on the specific structure of the random walk describing PageRank: P decomposes into a natural random walk matrix $D^{-1}A$, and a rank-1 matrix $\mathbf{1} \cdot r^T$, with strictly positive weights $(1 - \beta)$ and β respectively, cf. (1.1). This structure of P has two key implications. Jeh and Widom [20] and Haveliwala [19] observed a key linearity relation – the global PageRank vector is the average of the n personalized PageRank vectors corresponding to those obtained by setting $r = e_i$ for $1 \leq i \leq n$. That is, these n personalized PageRank vectors centered at each node form a basis for all personalized PageRank vectors, including the global PageRank. Therefore, the problem boils down to computing the personalized PageRank for a given node. Forgaras *et al.* [14] used the fact that for the personalized PageRank centered at a given node i (i.e., $r = e_i$), the associated random walk has probability β at every step to jump back to node i. This is a renewal time for the random walk and hence by the standard renewal theory, the estimate of a node's stationary probability (or personalized PageRank) can be obtained through enough samples of renewal cycles along with standard concentration results.

Subsequent to the above two key observations, Avrachenkov *et al.* [4] surveyed variants to Fogaras' random walk algorithm, such as computing the frequency of visits to nodes across the complete sample path rather than only the end node. Bahmani *et al.* [5] addressed how to incrementally update the PageRank vector for dynamically evolving graphs, or graphs where the edges arrive in a streaming manner. Das Sarma *et*

al. extended the algorithm to streaming graph models [30], and distributed computing models [31], "stitching" together short random walks to obtain longer samples, and thus reducing communication overhead. More recently, building on the same sets of observation, Borgs *et al.* [7] provided a sublinear time algorithm for estimating global PageRank using multi-scale matrix sampling. They use geometric-length random walk samples, but do not need to sample for all n personalized PageRank vectors. The algorithm returns a set of "important" nodes such that the set contains all nodes with PageRank greater than a given threshold, Δ, and does not contain any node with PageRank less than Δ/c with probability $1 - o(1)$, for a given $c > 3$. The algorithm runs in time $O\left(\frac{1}{\Delta}\text{polylog}(n)\right)$.

Andersen et al. designed a backward variant of these algorithms [2]. Previously, to compute the global pagerank of a specific node j, we would average over all personalized pagerank vectors. The algorithm proposed by Andersen et al. estimates the global pagerank of a node j by approximating the "contribution vector", i.e. estimating for the j^{th} coordinates of the personalized pagerank vectors that contribute the most to π_j.

These algorithms are effectively local: to compute the personalized PageRank of a node i (i.e., $r = e_i$), the sampled nodes are within distance distributed according to a geometric random variable with parameter β, which is on average $1/\beta$. The global PageRank computation requires personalized computation of other nodes as well but they can be performed asynchronously, in parallel. They allow for incremental updates as the network changes dynamically. However, all of these rely on the crucial property that the random walk has renewal time that is distributed geometrically with parameter $\beta > 0$ (that does not scale with network size n, like $1/n$, for example). This is because the transition matrix P decomposes as per (1.1) with $\beta \in (0, 1)$, not scaling (down) with n. In general, the transition matrix of any irreducible, aperiodic, positive-recurrent Markov chain will not have such a decomposition property (and hence known renewal

time), making the above algorithms inapplicable.

■ 1.3 Contributions

This thesis designs a local algorithm for estimating stationary distributions of general Markov chains. We develop a *truncated Monte Carlo* method, which answers the following question: for a given node i of a MC with a finite or countably infinite state-space, is the stationary probability of i larger or smaller than Δ, for a given threshold $\Delta \in (0, 1)$?

Our proposed randomized algorithm utilizes only the 'local' neighborhood of the node i, where neighborhood is defined with respect to the transition graph of the Markov chain, to produce an approximate answer to this question. The algorithm's computation (as well as 'radius' of local neighborhood) scales inversely proportional to Δ. The algorithm has an easy to verify stopping condition with provable performance guarantees. For nodes such that $\pi_i \geq \Delta$, we show in Theorem 4.2.1 that our estimate is within an $\epsilon Z_{\max}$ multiplicative factor of the true π_i, where $Z_{\max}$ is a function of the mixing properties of the graph. For nodes such that $\pi_i < \Delta$, we obtain an upper bound to label them as low probability nodes. Examples in Chapter 3 and Section 6.3 illustrate that *any* local algorithm cannot avoid the possibility of overestimating the stationary probabilities of nodes in settings where the graph mixes poorly.

The algorithm proposed is based on a basic property: the stationary probability of a node in a MC is the inverse of the average value of its "return time" under the MC. Therefore, one way to obtain a good estimate of stationary probability of a node is to sample return times to the node in the MC and use its empirical average to produce an estimate. To keep the algorithm "local", since return times can be arbitrary long, we truncated the sample return times at some threshold θ. In that sense, our algorithm is a *truncated Monte Carlo* method.

The optimal choice for the truncation parameter, the number of samples, and the

stopping conditions is far from obvious when one wishes (a) to not utilize any prior information about MC for generic applicability, and (b) to provide provable performance guarantees. The key contribution of this paper lies in resolving this challenge by means of simple rules.

In establishing correctness and convergence properties of the algorithm, we utilize the exponential concentration of return times in Markov chains. For finite state Markov chains, such a result follows from known results (see Aldous and Fill [1]). For countably infinite state space Markov chains, we build upon a result by Hajek [17] on the concentration of certain types of hitting times in order to prove that the return times to a node concentrate around its mean. We use these concentration results to upper bound the estimation error and the algorithm running time as a function of the truncation threshold θ and the mixing properties of the graph. For graphs that mix quickly, the distribution over return times concentrates more sharply around its mean, and therefore we obtain tighter performance guarantees.

■ 1.4 Outline

In Chapter 2, we give a brief overview of the fundamentals in Markov chains and probability theory needed to understand our results. In Chapter 3, we present the problem statement and our algorithm. In Chapter 4, we prove theoretical guarantees in both the finite state space and countably infinite state space settings. We show that the algorithm terminates in finite time, and with high probability, the estimate is an upper bound of the true value. Given specific properties of the Markov chain, we prove tighter concentration results for the tightness of our approximation as well as tighter bounds for the total running time. The tightness of our approximation will be limited by the mixing properties of the Markov chain. In Chapter 5, we propose an extension to the algorithm that allows us to reuse computation to estimate the stationary probabilities

of multiple nodes simultaneously. We similarly show guarantees for the tightness of our approximation as a function of the mixing properties of the Markov chain. In Chapter 6, we show simulation results of running our algorithm on specific MCs, verifying that the algorithm performs well in reasonable settings. We also show an example of a Markov chain that mixes poorly and discuss the limitations of our algorithm in that setting. Finally, we present closing remarks and discussion in Chapter 7.

Chapter 2

Preliminaries

2.1 Markov chains

For this thesis, we consider discrete-time, time-homogenous, countable state space Markov chains. A Markov chain with state space Σ and transition probability matrix P is a sequence of random variables $\{X_t\}$ for $t \in \{0\} \cup \mathbb{Z}_+$ such that

- For all t, $X_t \in \Sigma$.
- $P : \Sigma \times \Sigma \to [0, 1]$, and P is stochastic: For all $i \in \Sigma$,

$$\sum_{j \in \Sigma} P_{ij} = 1.$$

- $\{X_t\}$ satisfies the Markov property:

$$\mathbb{P}(X_{t+1} = j | X_0 = x_0, X_1 = x_1, \ldots X_t = i) = \mathbb{P}(X_{t+1} = j | X_t = i) = P_{ij}.$$

Let $P_{xy}^{(n)}$ denote the value of coordinate (x, y) of the matrix P^n.

Definition 2.1.1. *The Markov chain is* irreducible *if for all $x, y \in \Sigma$, there exists $a, b \in \mathbb{Z}_+$ such that*

$$P_{xy}^{(a)} > 0 \quad \textit{and} \quad P_{xy}^{(b)} > 0.$$

Definition 2.1.2. *The Markov chain is* aperiodic *if for all $x \in \Sigma$, there exists $n_0 \in \mathbb{Z}_+$ such that for all $n > n_0$,*

$$P_{xx}^{(n)} > 0.$$

The Markov chain can be visualized as a random walk over a weighted directed graph $G = (\Sigma, E)$, where Σ is the set of nodes, $E = \{(i,j) \in \Sigma \times \Sigma : P_{ij} > 0\}$ is the set of edges, and P describes the weights of the edges. We refer to G as the *Markov chain graph.* The state space Σ is assumed to be either finite or countably infinite. If it is finite, let $n = |\Sigma|$ denote the number of nodes in the network.[1] If the Markov chain is irreducible, it is equivalent to the graph being strongly connected (for all $x, y \in \Sigma$ there exists a path from x to y). If the Markov chain is aperiodic, it is equivalent to the statement that the graph cannot be partitioned into more than one partition such that there are no edges within each partition (for example, a bipartite graph is not aperiodic). The *local* neighborhood of node $i \in \Sigma$ within distance $r \geq 1$ is defined as $\{j \in \Sigma : d_G(i,j) \leq r\}$, where $d_G(i,j)$ is the length of the shortest directed path from i to j in G.

Throughout the paper, we will use the notation $\mathbb{E}_i[f(\{X_t\})] = \mathbb{E}[f(\{X_t\})|X_0 = i]$, and $\mathbb{E}_U[f(\{X_t\})]$ to denote the expectation of $f(\{X_t\})$ under the condition that X_0 is distributed uniformly over state space Σ (of course this is only well defined for a finite state space Markov chain). Similarly, we denote $\mathbb{P}_i(\text{event}) = \mathbb{P}(\text{event}|X_0 = i)$. Let the return time to a node i be defined as

$$T_i = \inf\{t \geq 1 \mid X_t = i\}, \tag{2.1}$$

[1]Throughout the paper, Markov chain and random walk on a network are used interchangeably; similarly nodes and states are used interchangeably. The stationary probability of a node quantifies the "importance" or "weight" of a node.

and let the maximal hitting time to a node i be defined as

$$H_i = \max_{j \in \Sigma} \mathbb{E}_j[T_i]. \tag{2.2}$$

The maximal hitting time may only be defined for finite state space Markov chains.

Definition 2.1.3. *A Markov chain is* recurrent *if for all* $i \in \Sigma$,

$$\mathbb{P}_i(T_i < \infty) = 1.$$

Definition 2.1.4. *A Markov chain is* positive recurrent *if for all* $i \in \Sigma$,

$$\mathbb{E}_i[T_i] < \infty.$$

We restrict ourselves to positive recurrent Markov chains. Since the expected return time is finite for all nodes, this means that the random walk cannot "drift to infinity". All irreducible finite Markov chains are positive recurrent, thus this property is only interesting in the countably infinite state space setting.

Definition 2.1.5. *A function* $\pi : \Sigma \to [0, 1]$ *is a stationary distribution for the Markov chain if*

- *For all* $i \in \Sigma$, $\pi_i = \sum_{j \in \Sigma} \pi_j P_{ji}$.
- $\sum_{i \in \Sigma} \pi_i = 1$.

In other words, if π describes a distribution over the state space, then after an arbitrary number of transitions in the Markov chain, the distribution remains the same. $\pi^T P^n = \pi^T$ for all n. If the Markov chain is irreducible and positive recurrent, then there exists a unique stationary distribution. Positive recurrence is equivalent to "stability", and the stationary distribution captures the equilibrium of the system.

It is well known [1, 26] that for any nodes $i, j \in \Sigma$,

$$\pi_i = \frac{1}{\mathbb{E}_i[T_i]}, \tag{2.3}$$

and

$$\pi_j = \frac{\mathbb{E}_i[\text{visits to node } j \text{ before time } T_i]}{\mathbb{E}_i[T_i]}. \tag{2.4}$$

The fundamental matrix Z of a finite state space Markov chain is defined as

$$Z = \sum_{t=0}^{\infty} \left(P^{(t)} - \mathbf{1}\pi^T \right) = \left(I - P + \mathbf{1}\pi^T \right)^{-1}.$$

i.e., the entries of the fundamental matrix Z are defined by

$$Z_{ij} = \sum_{t=0}^{\infty} \left(P_{ij}^{(t)} - \pi_j \right).$$

Entry Z_{ij} corresponds to how quickly a random walk beginning at node i mixes to node j. We restate here a few properties shown in Aldous and Fill [1] which will be useful for our analysis.

Lemma 2.1.6. *For $i \neq j$,*

$$\mathbb{E}_i[T_j] = \frac{Z_{jj} - Z_{ij}}{\pi_j}.$$

Lemma 2.1.7. *For distinct $i, k \in \Sigma$, the expected number of visits to a node j beginning from node k before time T_i is equal to*

$$\mathbb{E}_k \left[\sum_{t=1}^{\infty} \mathbf{1}_{\{X_t = j\}} \mathbf{1}_{\{t \leq T_i\}} \right] = \pi_j (E_k[T_i] + E_i[T_j] - E_k[T_j]).$$

■ 2.2 Foster-Lyapunov analysis

Foster introduced a set of techniques for analyzing countable state space Markov chains. [15]

Theorem 2.2.1. *Let $\{X_t\}$ be a discrete time, irreducible, aperiodic Markov chain on countable state space Σ with transition probability matrix P. $\{X_t\}$ is positive recurrent if and only if there exists a Lyapunov function $V : \Sigma \to \mathbb{R}_+$, $\gamma > 0$ and $b \geq 0$, such that*

1. *For all $x \in \Sigma$,*

$$\mathbb{E}\left[V(X_{t+1})|X_t = x\right] \leq \infty,$$

2. *For all $x \in \Sigma$ such that $V(x) > b$,*

$$\mathbb{E}\left[V(X_{t+1}) - V(X_t)|X_t = x\right] \leq -\gamma.$$

In words, given a positive recurrent Markov chain, there exists a Lyapunov function $V : \Sigma \to \mathbb{R}_+$ and a decomposition of the state space into $B = \{x \in \Sigma : V(x) \leq b\}$ and $B^c = \{x \in \Sigma : V(x) > b\}$ such that there is a uniform negative drift in B^c towards B and $|B|$ is finite.

In fact, for any irreducible, aperiodic, Markov chain, the following function is a valid Lyapunov function. Choose any $i \in \Sigma$.

$$V(x) = \mathbb{E}[T_i|X_0 = x]$$

where $T_i = \inf\{t \geq 0 : X_t = i\}$. Thus $B = \{i\}$. And $V(i) = 0$. For all $x \in \Sigma \setminus \{i\}$,

$$V(x) = 1 + \sum_{y \in \Sigma} P_{xy} V(y).$$

And thus,

$$\sum_{y \in \Sigma} P_{xy} V(y) - V(x) = -1.$$

A Lyapunov function for a positive recurrent Markov chain helps to impose a natural ordering over the state space that allows us to prove properties of the Markov chain. There have been many results following the introduction of the Foster-Lyapunov Criteria, which give bounds on the stationary probabilities, return times, and distribution of return times as a function of the Lyapunov function [6, 17]. We present a result from Hajek that will be useful in our analysis.

Theorem 2.2.2 (Hajek 1982). *[17] Let $\{X_t\}$ be an irreducible, aperiodic, positive recurrent Markov chain on a countable state space Σ with transition probability matrix P. Assume that there exists a Lyapunov function $V : \Sigma \to \mathbb{R}_+$ and values $\nu_{\max}, \gamma > 0$, and $b \geq 0$ such that*

1. *The set $B = \{x \in \Sigma : V(x) \leq b\}$ is finite,*

2. *For all $x, y \in \Sigma$ such that $\mathbb{P}(X_{t+1} = j | X_t = i) > 0$,*

$$|V(j) - V(i)| \leq \nu_{\max},$$

3. *For all $x \in \Sigma$ such that $V(x) > b$,*

$$\mathbb{E}[V(X_{t+1}) - V(X_t) | X_t = x] \leq -\gamma.$$

Let the random variable $\tau_B = \inf\{t : X_t \in B\}$. Then for any x such that $V(x) > b$, and for any choice of constants $\omega > 0$, η, ρ, and λ satisfying

$$0 < \eta \leq \min\left(\omega, \frac{\gamma\omega^2}{e^{\omega\nu_{\max}} - (1 + \omega\nu_{\max})}\right),$$

$$\rho = 1 - \gamma\eta + \frac{(e^{\omega\nu_{\max}} - (1 + \omega\nu_{\max}))\,\eta^2}{\omega^2},$$

$$0 < \lambda < \ln(\frac{1}{\rho}),$$

the following two inequalities hold:

$$\mathbb{P}[\tau_B > k | X_0 = x] \leq e^{\eta(V(x)-b)}\rho^k,$$

$$\mathbb{E}[e^{\lambda\tau_B} | X_0 = x] \leq e^{\eta(V(x)-b)}\left(\frac{e^\lambda - 1}{1 - \rho e^\lambda}\right) + 1.$$

A concrete set of constants that satisfy the conditions above are

$$\omega = \frac{1}{\nu_{\max}}, \eta = \frac{\gamma}{2(e-2)\nu_{\max}^2}, \text{ and } \rho = 1 - \frac{\gamma^2}{4(e-2)\nu_{\max}^2}. \tag{2.5}$$

■ 2.3 Concentration of random variables

In this section, we state fundamental theorems for the concentration of a sum of i.i.d. random variables.

Theorem 2.3.1 (Chernoff's Multiplicative Bound for Binomials).
Let $\{X_1, X_2, X_3, \ldots X_N\}$ *be a sequence of independent identically distributed Bernoulli random variables, such that for all* i, $X_i = 1$ *with probability* p *and* $X_i = 0$ *otherwise. Then for any* $\epsilon > 0$,

$$\mathbb{P}\left(\left|\frac{1}{N}\sum_{i=1}^{N} X_i - \mathbb{E}[X]\right| \geq \epsilon\mathbb{E}[X]\right) \leq \left(\frac{e^\epsilon}{(1+\epsilon)^{(1+\epsilon)}}\right)^{Np} + \left(\frac{e^{-\epsilon}}{(1-\epsilon)^{(1-\epsilon)}}\right)^{Np}$$
$$\leq 2e^{-\frac{\epsilon^2 Np}{3}}.$$

Theorem 2.3.2 (Chernoff's Multiplicative Bound for Bounded Variables).
Let $\{X_1, X_2, X_3, \ldots X_N\}$ *be a sequence of independent identically distributed strictly bounded random variables, such that* $X_i \sim X$ *for all* i, *and* $X \in [0, \theta]$. *Then for any*

$\epsilon > 0$,

$$\mathbb{P}\left(\left|\frac{1}{N}\sum_{i=1}^{N} X_i - \mathbb{E}[X]\right| \geq \epsilon\mathbb{E}[X]\right) \leq \left(\frac{e^{\epsilon}}{(1+\epsilon)^{(1+\epsilon)}}\right)^{\frac{N\mathbb{E}[X]}{\theta}} + \left(\frac{e^{-\epsilon}}{(1-\epsilon)^{(1-\epsilon)}}\right)^{\frac{N\mathbb{E}[X]}{\theta}}$$
$$\leq 2e^{-\frac{\epsilon^2 N\mathbb{E}[X]}{3\theta}}.$$

■ 2.4 Monte Carlo Methods for Pagerank

In this section, we present the key concept behind using Monte Carlo methods to approximate Pagerank. These techniques sample short random paths, and approximate PageRank with either the frequency of visits along the paths, or the end node of the paths. Given a directed network $G = (V, E)$ of n nodes, let A denote the adjacency matrix, and let D denote the out degree diagonal matrix. $A_{ij} = 1$ if $(i, j) \in E$ and 0 otherwise. D_{ii} is the out degree of node i, and $D_{ij} = 0$ for $i \neq j$. Let e_i denote the vector such that the i^{th} coordinate is 1 and all other coordinates are 0. The PageRank vector π is given by the stationary distribution of the Markov chain having transition matrix P given by

$$P = (1 - \beta)D^{-1}A + \beta\mathbf{1} \cdot r^T. \tag{2.6}$$

Here $\beta \in (0, 1)$ is a fixed scalar, and r is a fixed probability vector over the n nodes. Thus, the PageRank vector π satisfies

$$\pi^T = \pi^T P = (1 - \beta)\pi^T D^{-1}A + \beta r^T \tag{2.7}$$

where $\pi^T \cdot \mathbf{1} = 1$. We denote the personalized pagerank vector for seed vector r with $PPV(r)$. Haveliwala showed that personalized PageRank vectors respect linearity. [19]

$$PPV(\alpha_1 \cdot r_1 + \alpha_2 \cdot r_2) = \alpha_1 \cdot PPV(r_1) + \alpha_2 \cdot PPV(r_2).$$

Therefore, we only need to find the personalized PageRank vectors for a set of basis vectors. These local algorithms calculate the personalized pagerank vectors where $r = e_i$. The key insight to these algorithms is that the random walk "resets" to distribution r every time it takes a β jump. Therefore Jeh and Widom showed that the distribution over the end node of a geometrically-distributed-length random walk is equivalent to the personalized pagerank vector starting from the beginning node [20]. The proof is shown here to highlight this critical property of PageRank.

Theorem 2.4.1. *Let G be a geometrically distributed random variable with parameter β such that $\mathbb{P}(G = t) = \beta(1-\beta)^t$ for $t \in \{0, 1, 2, \dots\}$. Consider a simple random walk that starts at node i and takes G steps according to a transition probability matrix $D^{-1}A$. Then the j^{th} coordinate of vector $PPV(e_i)$ (denote by $PPV_j(e_i)$) is*

$$PPV_j(e_i) = \mathbb{P}(\text{the random walk ends at node } j).$$

In addition,

$$PPV_j(e_i) = \frac{\mathbb{E}[\#\text{ of visits to node } j]}{\mathbb{E}[G]}.$$

Proof. By definition,

$$PPV(e_i) = (1-\beta)PPV(e_i)D^{-1}A + \beta r^T.$$

Therefore,

$$\begin{aligned}
PPV_j(e_i) &= \beta e_i^T (I - (1-\beta)D^{-1}A)^{-1} e_j \\
&= \beta e_i^T \sum_{t=0}^{\infty} (1-\beta)^t (D^{-1}A)^t e_j \\
&= \sum_{t=0}^{\infty} \mathbb{P}(G = t) e_i^T (D^{-1}A)^t e_j \\
&= \sum_{t=0}^{\infty} \mathbb{P}(G = t) \mathbb{P}(\text{random walk visits node } j \text{ at time } t) \\
&= \mathbb{P}(\text{the random walk ends at node } j).
\end{aligned}$$

Similarly,

$$\begin{aligned}
PPV_j(e_i) &= \beta e_i^T \sum_{t=0}^{\infty} (1-\beta)^t (D^{-1}A)^t e_j \\
&= \beta^2 e_i^T \sum_{t=0}^{\infty} (D^{-1}A)^t \sum_{g=t}^{\infty} (1-\beta)^g e_j \\
&= \beta^2 e_i^T \sum_{g=0}^{\infty} \sum_{t=0}^{g} (D^{-1}A)^t (1-\beta)^g e_j \\
&= \beta^2 \sum_{g=0}^{\infty} (1-\beta)^g \sum_{t=0}^{g} e_i^T (D^{-1}A)^t e_j \\
&= \beta^2 \sum_{g=0}^{\infty} (1-\beta)^g \sum_{t=0}^{g} \mathbb{P}(\text{random walk visits node } j \text{ at time } t) \\
&= \beta \sum_{g=0}^{\infty} \beta (1-\beta)^g \mathbb{E}[\#\text{ of visits to node } j \text{ for length } t \text{ random walk}] \\
&= \frac{\sum_{g=0}^{\infty} \mathbb{P}(G = g) \mathbb{E}[\#\text{ of visits to node } j \text{ for length } t \text{ random walk}]}{\mathbb{E}[G]} \\
&= \frac{\mathbb{E}[\#\text{ of visits to node } j]}{\mathbb{E}[G]}.
\end{aligned}$$

■

For each page i calculate N independent geometric-length random walks. Approximate $PPV_j(e_i)$ with the empirical distribution of the end nodes of these random walks. By standard concentration results of Binomial random variables, the empirical distribution concentrates around $PPV(e_i)$. Alternatively, we can approximate $PPV_j(e_i)$ with the fraction of visits to node j along the paths of these random walks. Again, by standard concentration results of Binomial random variables, the estimate will concentrate around $PPV_j(e_i)$.

Chapter 3

Algorithm

■ 3.1 Problem Statement

Consider a discrete time, irreducible, aperiodic, positive recurrent Markov chain $\{X_t\}_{t\geq 0}$ on a countable state space Σ having transition probability matrix $P : \Sigma \times \Sigma \to [0, 1]$. Remember that all irreducible finite Markov chains are positive recurrent. Given a node i and threshold $\Delta > 0$, we would like to know whether $\pi_i \geq \Delta$ or $\pi_i < \Delta$. Our goal is to achieve this using the transition probabilities of edges within a local neighborhood of i. The local neighborhood is defined as all nodes within distance θ of i. We would like to keep θ small while still obtaining good estimates.

First we show an example to illustrate the limitations of local algorithms. Consider the graph shown in Figure 3.1. It is a graph with n nodes, composed of a size k clique

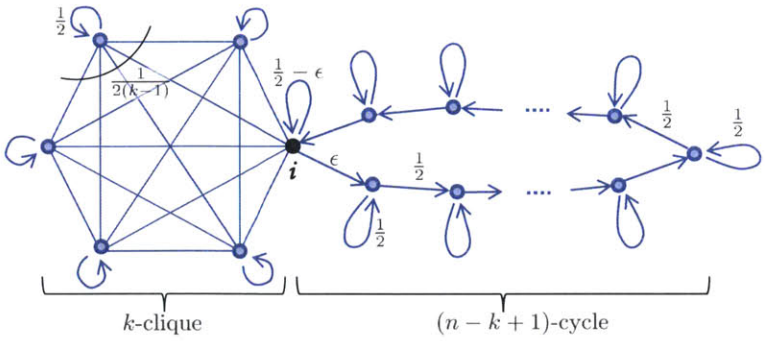

Figure 3.1. Graph of a clique plus loop

connected to a size $(n-k+1)$ cycle. Let A denote the set of nodes in the k-clique. For all nodes $j \in \Sigma \setminus i$, let the self loop probability $P_{jj} = \frac{1}{2}$. For node j in the clique excluding i, let the remaining transition probabilities be uniform over the neighbors. Thus with probability $\frac{1}{2}$, a random walk stays at node j, but with probability $\frac{1}{2}$ the random walk chooses a random neighbor. For node j in the loop, if the random walk does not remain at node j with probability $\frac{1}{2}$, then the random walk travels counterclockwise to the subsequent node in the loop. For node i, with probability ϵ the random walk enters the loop, and with probability $\frac{1}{2}$ the random walk chooses any neighbor in the clique.

We can show that the expected return time to node i is

$$\begin{aligned}\mathbb{E}[T_i] &= \left(\frac{1}{2} - \epsilon\right) + \epsilon\ \mathbb{E}[T_i|\text{first step enters cycle}] + \frac{1}{2}\mathbb{E}[T_i|\text{first step enters clique}] \\ &= 1 + \epsilon(2(n-k)) + \frac{1}{2}(2(k-1)) = k + 2\epsilon(n-k) \end{aligned} \tag{3.1}$$

Therefore, keeping k, ϵ fixed, as n increases, $\pi_1 = 1/\mathbb{E}[T_1]$ decreases and scales as $O(1/n)$. For any fixed threshold θ, n can be chosen such that $n \gg \theta + k$. Any algorithm that only has access to a local neighborhood of node i up to distance θ will not be able to obtain an accurate estimate of π_i, as it cannot differentiate between a graph having $n = \theta + k$ or $n \gg \theta + k$ nodes. This shows a setting in which any local algorithm will perform poorly. For another example, see the Magnet graph in Section 6. In these examples, for any local algorithm, we can at best provide an estimate that is an upper bound for π_i. We cannot guarantee that the estimate is close, as the true value for π_i can be arbitrarily small as a function of the graph properties outside of the local neighborhood. Thus, it is impossible to have a local algorithm that is always able to determine $\pi_i < \Delta$ or $\pi_i \geq \Delta$ accurately for all nodes $i \in \Sigma$.

Revised Problem Statement. Therefore, we consider the next natural variation of the above stated problem and answer it successfully in this thesis: for nodes with $\pi_i \geq \Delta$,

correctly determine that $\pi_i \geq \Delta$ with high probability (with respect to randomness in the algorithm). Equivalently, when the algorithm declares that $\pi_i < \Delta$, then node i has $\pi_i < \Delta$ with high probability. Note that the algorithm may incorrectly guess that $\pi_i \geq \Delta$ for nodes that have $\pi_i < \Delta$.

As discussed, the example in Figure 3.1 shows that such one-sided error is unavoidable for *any* local algorithm. In that sense, we provide the best possible local solution for computing stationary distribution.

■ 3.2 Monte Carlo Return Time (MCRT) Algorithm

Algorithm. Given a threshold $\Delta \in (0, 1)$ and a node $i \in \Sigma$, the algorithm outputs an estimate $\hat{\pi}_i$ of π_i. The algorithm relies on the relation $\pi_i = 1/\mathbb{E}_i[T_i]$ (cf. Eq 2.3), and estimates π_i by sampling from $\min(T_i, \theta)$ for some θ large enough. By definition, $\mathbb{E}_i[T_i]$ is equal to the expected length of a random walk that starts at node i and stops the moment it returns to node i. Since the length of T_i is possibly unbounded, the algorithm truncates the samples of T_i at some threshold θ. Therefore, the algorithm involves taking N independent samples of a truncated random walk beginning at node i and stopping either when the random walk returns to node i, or when the length exceeds θ. Each sample is generated by simulating the random walk using "crawl" operations over the Markov chain graph G.

As $N \to \infty$ and $\theta \to \infty$, the estimate will converge almost surely to π_i, due to the strong law of large numbers and positive recurrence of the Markov chain (along with property (2.3)). The question remains how to choose θ and N to guarantee that our estimate will have a given accuracy with high probability. The number of samples N must be large enough to guarantee that the sample mean concentrates around the true mean of the random variable. We use Chernoff's bound (see Appendix 2.3.2) to choose a sufficiently large N, which increases with θ. Choosing θ is not as straightforward. If

θ is too small, then most of the samples may be truncated, and the sample mean will be far from the true mean. However, using a large θ requires also a large number of samples N, and may be unnecessarily expensive.

Since the algorithm has no prior knowledge about the distribution of T_i, we search for an appropriate θ by beginning with small values for θ, and increasing the value geometrically. This is order-optimal: the final choice of θ will be at most a constant factor more than the optimal value. Let the computation time be a function $f(\theta)$. Then the total cost of executing the algorithm repeatedly for geometrically increasing values of θ will be

$$\sum_{t=1}^{\log \theta} f(2^t).$$

If $f(x)$ is super-linear in x, then

$$\begin{aligned}\sum_{t=1}^{\log \theta} f(2^t) &= \sum_{t=1}^{\log \theta} \frac{2^t f(2^t)}{2^t} \\ &\leq \frac{f(\theta)}{\theta} \sum_{t=1}^{\log \theta} 2^t \\ &\leq \frac{f(\theta) 2\theta}{\theta} = 2f(\theta).\end{aligned}$$

Therfore, since the computation time of our algorithm for a fixed θ is super-linear in θ, repeating our algorithm for geometrically increasing values of θ until we find the best θ will cost at most 2 times the computation time of the algorithm given the optimal value for θ.

Monte Carlo Return Time (MCRT) Algorithm

Initialize Choose parameters Δ = threshold for importance, ϵ = closeness of the

estimate, and α = probability of failure. Set

$$t = 1, \theta^{(1)} = 2, N^{(1)} = \left\lceil \frac{6(1+\epsilon)\ln(8/\alpha)}{\epsilon^2} \right\rceil.$$

Step 1 (Gather Samples) For each k in $\{1, 2, 3, \ldots N^{(t)}\}$, generate independent samples s_k of $\min(T_i, \theta^{(t)})$. Let $p^{(t)}$ = fraction of samples that were truncated at $\theta^{(t)}$, and let

$$\hat{T}_i^{(t)} = \frac{1}{N^{(t)}} \sum_{k=1}^{N^{(t)}} s_k.$$

Step 2 (Termination Conditions) If either

$$\textbf{(a)} \quad \hat{T}_i^{(t)} > \frac{(1+\epsilon)}{\Delta} \qquad \text{or} \qquad \textbf{(b)} \quad \frac{p^{(t)}}{\hat{T}_i^{(t)}} < \epsilon\Delta,$$

then stop and return estimate

$$\hat{\pi}_i = \frac{1}{\hat{T}_i^{(t)}}.$$

Step 3 (Update Rules) Set

$$\theta^{(t+1)} \leftarrow 2 \cdot \theta^{(t)}, N^{(t+1)} \leftarrow \left\lceil \frac{3(1+\epsilon)\theta^{(t+1)}\ln(4\theta^{(t+1)}/\alpha)}{\hat{T}_i^{(t)}\epsilon^2} \right\rceil, \text{ and } t \leftarrow t+1.$$

Return to **Step 1**.

Throughout the paper, we will refer to the total number of iterations used in the algorithm as the value for t at the time of termination. One iteration in the algorithm refers to one round of executing Steps 1-3. The total number of "steps" taken by the algorithm is equivalent to the number of neighbor queries on the network. Therefore, it is the sum over all lengths of the random walk sample paths. The total number of steps taken within the first t iterations is $\sum_{k=1}^{t} N^{(t)}\hat{T}_i^{(t)}$. This is used to analyze the computation time of the algorithm.

The algorithm will terminate at stopping condition (a) when the estimate is smaller than $\frac{\Delta}{1+\epsilon}$. In Lemma 4.1.3, we prove that with high probability, $\hat{pi}_i \geq \frac{\pi_i}{1+\epsilon}$ in all iterations. Therefore, when the algorithm terminates at stopping condition (a), we know that $\pi_i \leq \Delta$ with high probability. Furthermore, important nodes such that $\pi_i \gg \Delta$ will not terminate at stopping condition (a) with high probability. Stopping condition (b) is chosen such that the algorithm terminates when $\hat{p}$ is small, i.e. when very few of the sample paths have been truncated at $\theta^{(t)}$. We will prove in Theorem 4.1.1(2) that this stopping condition will eventually be satisfied for all nodes when $\theta^{(t)} \geq \frac{1}{\epsilon}\Delta$. In fact, we will also prove that $\mathbb{P}(T_i > \theta^{(t)})$ decays exponentially in $\theta^{(t)}$, and thus the algorithm will terminate earlier at stopping condition (b) depending how quickly $\mathbb{P}(T_i > \theta^{(t)})$ decays. We show in Theorems 4.2.1 and 4.3.2 that when stopping condition (b) is satisfied, the expected error between the estimate and π_i is small.

■ 3.3 Bias correction

Our algorithm introduces a bias, as the expected truncated return time $\mathbb{E}[\min(T_i, \theta)]$ is always less than the true expected return time $\mathbb{E}[\hat{T}_i]$. We can partially correct for this bias by multiplying the estimate by $(1 - \hat{p}^{(t)})$. Let $\hat{\pi}_i^C$ denote the estimate after bias correction.

$$\hat{\pi}_i^C = \frac{(1 - \hat{p}^{(t)})}{\hat{T}_i^{(t)}}.$$

For graphs that mix quickly, this correction will do well, as we will show in following analyses and simulations.

Chapter 4

Theoretical Guarantees

4.1 Main Theorem

We state the main result establishing the correctness and convergence properties of the algorithm, and we provide an explicit bound on the total computation performed by the algorithm. We establish that the algorithm always stops in finite time. When the algorithm stops, if it outputs $\hat{\pi}_i < \Delta/(1+\epsilon)$, then with high probability $\pi_i < \Delta$. That is, when $\pi_i \geq \Delta$, the algorithm will output $\hat{\pi}_i \geq \Delta/(1+\epsilon)$ with high probability. Thus, if we use the algorithm to identify high probability nodes with $\pi_i \geq \Delta$, then it will never have false negatives, meaning it will never fail to identify nodes that are truly important (with high probability). However, our algorithm may possibly produce false positives, identifying nodes as "high probability" even when in fact they may not be. Effectively, the false positives are nodes that are "high probability" nodes with respect to their local neighborhood, but not globally. In that sense, the false positives that the algorithm produces are not arbitrary false positives, but nodes that have high probability, "locally". As showed in the example in Chapter 3, it is impossible for any finite, local algorithm to avoid false positives entirely. In that sense, ours is as good a local algorithm as one can expect to have for general Markov chains. The following statement holds in general for all positive recurrent countable state space Markov chains. It shows a worst case dependence on the parameters chosen for the

algorithm and irregardless of the properties of the graph.

Theorem 4.1.1. *For an aperiodic, irreducible, positive recurrent, countable state space Markov chain, for any* $i \in \Sigma$*:*

1. **Correctness.** *When the algorithm (cf. Section 3.2) outputs* $\hat{\pi}_i < \Delta/(1+\epsilon)$*, then indeed* $\pi_i < \Delta$ *with probability at least* $(1-\alpha)$*. Equivalently, when* $\pi_i \geq \Delta$*, the algorithm outputs* $\hat{\pi}_i > \Delta/(1+\epsilon)$ *with probability at least* $(1-\alpha)$*.*

2. **Convergence.** *With probability 1, the number of iterations* t *in the algorithm is bounded above by*[1]

$$t \leq \ln\left(\frac{1}{\epsilon\Delta}\right),$$

and the total number of steps (or neighbor queries) used by the algorithm is bounded above by

$$\tilde{O}\left(\frac{\ln(\frac{1}{\alpha})}{\epsilon^3\Delta}\right).$$

The Markov chain is required to be aperiodic, irreducible, and positive recurrent so that there exists an unique stationary distribution. All irreducible finite state space Markov chains are positive recurrent, therefore the positive recurrence property is only explicity required and used in the analysis of countably infinite state space Markov chains. Part 1 of this theorem asserts that the estimate is an upper bound with high probability. Part 2 of this theorem asserts that the algorithm terminates in finite time as a function of the parameters of the algorithm. Both of these results are independent from specific properties of the Markov chain. In fact, we can obtain tighter characterizations (see sections 4.2 and 4.3) of both the correctness and the running time as functions of the properties of the graph. The analysis depends on how sharply the distribution over return times concentrate around the mean.

[1] We use the notation $\tilde{O}(f(a)g(b))$ to mean $\tilde{O}(f(a))\tilde{O}(g(b)) = \tilde{O}(f(a)\text{polylog}f(a))\tilde{O}(g(b)\text{polylog}g(b))$.

Proof of Theorem 4.1.1(1). We prove that for any node such that $\pi_i > \Delta$, the algorithm will output $\hat{\pi}_i > \Delta/(1+\epsilon)$ with probability at least $(1-\alpha)$.

First, we prove that $\hat{T}_i^{(t)}$ lies within an ϵ multiplicative interval around $\mathbb{E}_i[\hat{T}_i^{(t)}]$ with high probability.

Lemma 4.1.2. *For every $t \in \mathbb{Z}_+$,*

$$\mathbb{P}_i\left(\bigcap_{k=1}^{t}\left\{\left|\hat{T}_i^{(k)} - \mathbb{E}_i\left[\hat{T}_i^{(k)}\right]\right| \leq \epsilon \mathbb{E}_i\left[\hat{T}_i^{(k)}\right]\right\}\right) > 1-\alpha.$$

Proof. Let A_t denote the event $\left\{\left|\hat{T}_i^{(t)} - \mathbb{E}_i\left[\hat{T}_i^{(t)}\right]\right| \leq \epsilon \mathbb{E}_i\left[\hat{T}_i^{(t)}\right]\right\}$. Since $N^{(t)}$ is a random variable due to its dependence on $\hat{T}_i^{(t-1)}$, the distribution of $\hat{T}_i^{(t)}$ depends on the value of $\hat{T}_i^{(t-1)}$. Thus, we condition on the occurrence of A_{t-1}, such that

$$\begin{aligned} N^{(t)} &= \left\lceil \frac{3(1+\epsilon)\theta^{(t)} \ln(4\theta^{(t)}/\alpha)}{\hat{T}_i^{(t-1)}\epsilon^2} \right\rceil \\ &\geq \frac{3(1+\epsilon)\theta^{(t)} \ln(4\theta^{(t)}/\alpha)}{(1+\epsilon)\mathbb{E}_i[\hat{T}_i^{(t-1)}]\epsilon^2} = \frac{3\theta^{(t)} \ln(4\theta^{(t)}/\alpha)}{\mathbb{E}_i[\hat{T}_i^{(t-1)}]\epsilon^2}. \end{aligned}$$

Then we apply Chernoff's bound for independent identically distributed bounded random variables (see Theorem 2.3.2), substitute in for $N^{(t)}$, and use the facts that $\mathbb{E}_i[\hat{T}_i^{(t)}] \geq \mathbb{E}_i[\hat{T}_i^{(t-1)}]$ and $\theta^{(t)} = 2^t$ for all t, so

$$\begin{aligned} \mathbb{P}_i\left(A_t | A_{t-1}\right) &\geq 1 - 2\exp\left(-\frac{\epsilon^2 N^{(t)} \mathbb{E}_i[\hat{T}_i^{(t)}]}{3\theta^{(t)}}\right) \\ &\geq 1 - 2\exp\left(-\frac{\mathbb{E}_i[\hat{T}_i^{(t)}] \ln(4\theta^{(t)}/\alpha)}{\mathbb{E}_i[\hat{T}_i^{(t-1)}]}\right) \\ &\geq 1 - \frac{\alpha}{2\theta^{(t)}} \geq 1 - \frac{\alpha}{2^{t+1}}. \end{aligned}$$

It can be verified that $\mathbb{P}_i(A_1)$ is similarly lower bounded by $1 - \frac{\alpha}{4}$ using the definition

of $N^{(1)}$. Therefore,

$$\mathbb{P}_i\left(\bigcap_{k=1}^{t} A_k\right) = \mathbb{P}_i\left(A_1\right)\prod_{k=2}^{t}\mathbb{P}_i\left(A_t|A_{t-1}\right) \geq \prod_{k=1}^{t}\left(1-\frac{\alpha}{2^{k+1}}\right).$$

We use the fact that $\prod_{k=1}^{t}(1-x_k) \geq 1-\sum_{k=1}^{t} x_k$ when $\sum_{k=1}^{t} x_k \leq 1$, to show that

$$\mathbb{P}_i\left(\bigcap_{k=1}^{t} A_k\right) \geq 1-\sum_{k=1}^{t}\frac{\alpha}{2^k} = 1-\alpha(1-2^{-t}) \geq 1-\alpha.$$

■

Next, we proceed to prove the main correctness result.

Lemma 4.1.3 (Correctness). *When the algorithm outputs $\hat{\pi}_i < \Delta/(1+\epsilon)$, then indeed $\pi_i < \Delta$ with probability at least $(1-\alpha)$. Equivalently, when $\pi_i \geq \Delta$, the algorithm outputs $\hat{\pi}_i > \Delta/(1+\epsilon)$ with probability at least $(1-\alpha)$.*

Proof. By Lemma 4.1.2, for all t, with probability $(1-\alpha)$,

$$\hat{T}_i^{(t)} \leq (1+\epsilon)\mathbb{E}_i[\hat{T}_i^{(t)}].$$

By definition of $\mathbb{E}_i[\hat{T}_i^{(t)}]$,

$$\mathbb{E}_i[\hat{T}_i^{(t)}] \leq \mathbb{E}_i[T_i].$$

Therefore,

$$\hat{\pi}_i \geq \frac{\pi_i}{1+\epsilon}. \tag{4.1}$$

If $\hat{\pi}_i < \Delta/(1+\epsilon)$, then

$$\frac{\pi_i}{1+\epsilon} < \frac{\Delta}{1+\epsilon} \implies \pi_i < \Delta.$$

■

Proof of Theorem 4.1.1(2). We provide a strict upper bound on the running time of the algorithm for all nodes $i \in \Sigma$.

First, in order to analyze the total computation time of the algorithm, we prove that the total number of steps taken by the algorithm within the first t iterations scales with 2^t.

Lemma 4.1.4. *With probability greater than $(1-\alpha)$, the total number of steps taken by the algorithm within the first t iterations is bounded by*

$$\tilde{O}\left(\frac{ln(\frac{1}{\alpha})2^t}{\epsilon^2}\right).$$

Proof. The total number of random walk steps (i.e., neighbor queries of the network) used in the algorithm over all t iterations is equal to $\sum_{k=1}^{t} N^{(k)}\hat{T}_i^{(k)}$. By Lemma 4.1.2, with probability greater than $(1-\alpha)$,

$$\frac{\hat{T}_i^{(t)}}{\hat{T}_i^{(t-1)}} \leq \frac{(1+\epsilon)\mathbb{E}_i[\hat{T}_i^{(t)}]}{(1-\epsilon)\mathbb{E}_i[\hat{T}_i^{(t-1)}]}$$

for all t. Because $\theta^{(t)}$ at most doubles in each iteration, $\frac{\mathbb{E}_i[\hat{T}_i^{(t)}]}{\mathbb{E}_i[\hat{T}_i^{(t-1)}]} \leq 2$. Therefore,

$$\begin{aligned}
\sum_{k=0}^{t} N^{(k)}\hat{T}_i^{(k)} &\leq \sum_{k=0}^{t} \frac{3(1+\epsilon)\theta^{(k)}\ln(4\theta^{(k)}/\alpha)\hat{T}_i^{(k)}}{\epsilon^2\hat{T}_i^{(k-1)}} \\
&\leq \sum_{k=0}^{t} \frac{6(1+\epsilon)^2\theta^{(k)}\ln(4\theta^{(k)}/\alpha)}{(1-\epsilon)\epsilon^2} \\
&\leq \frac{6(1+\epsilon)^2}{(1-\epsilon)\epsilon^2} O(t2^t + \ln(1/\alpha)2^t) \leq O\left(\frac{(1+\epsilon)^2\ln(1/\alpha)t2^t}{(1-\epsilon)\epsilon^2}\right).
\end{aligned}$$

We suppress the insignificant factors $(1+\epsilon)$ and $(1-\epsilon)$ and the polylogarithmic factors,

to show that the total number of steps taken by the algorithm is bounded above by

$$\tilde{O}\left(\frac{\ln(\frac{1}{\alpha})2^t}{\epsilon^2}\right).$$

■

Next we procede to prove the main convergence result.

Lemma 4.1.5 (Convergence). *The number of iterations t is always bounded above by*

$$t \leq \ln\left(\frac{1}{\epsilon\Delta}\right),$$

and with probability greater than $(1-\alpha)$, the total number of steps (or neighbor queries) used by the algorithm is bounded above by

$$\tilde{O}\left(\frac{\ln(\frac{1}{\alpha})}{\epsilon^3\Delta}\right).$$

Proof. By definition, $\hat{T}_i^{(t)} > \hat{p}^{(t)}\theta^{(t)}$. Thus,

$$\hat{T}_i^{(t)} > \hat{p}^{(t)}\theta^{(t)} \implies \frac{\hat{p}^{(t)}}{\hat{T}_i^{(t)}} < \frac{1}{\theta^{(t)}}.$$

And thus,

$$\theta^{(t)} > \frac{1}{\epsilon\Delta} \implies \frac{\hat{p}^{(t)}}{\hat{T}_i^{(t)}} < \epsilon\Delta.$$

Therefore for $\theta^{(t)} > \frac{1}{\epsilon\Delta}$, stopping condition (b) will be true and the algorithm will terminate. Recall that $\theta^{(t)} = 2^t$. Therefore the maximum number of iterations t before termination is bounded above by

$$t \leq \ln\left(\frac{1}{\epsilon\Delta}\right).$$

Using Lemma 4.1.4, the total number of steps (or neighbor queries) used by the algorithm is bounded above by

$$\tilde{O}\left(\frac{\ln(\frac{1}{\alpha})}{\epsilon^3\Delta}\right).$$

■

■ 4.2 Finite State Space Markov Chain

Using the properties of the graph, we can prove tighter bounds on the correctness and convergence of the algorithm. In this section, we present the results for finite state space Markov chains.

Correctness. The following theorem proves an upper bound on the looseness of approximation.

Theorem 4.2.1. *For a finite state space, irreducible, aperiodic Markov chain* $\{X_t\}$ *with state space* Σ *and transition probability matrix* P, *and for any* $i \in \Sigma$,

$$\begin{aligned}\frac{1}{\mathbb{E}_i[\hat{T}_i^{(t)}]} - \pi_i &= \frac{\mathbb{P}_i\left(T_i^{(t)} > \theta^{(t)}\right)}{\mathbb{E}_i[\hat{T}_i^{(t)}]}\left(Z_{ii} - \sum_{k\in\Sigma\setminus\{i\}} \mathbb{P}_i\left(X_{\theta^{(t)}} = k \middle| T_i > \theta^{(t)}\right) Z_{ki}\right) \\ &\leq \frac{\pi_i}{\mathbb{E}_i[\hat{T}_i^{(t)}]}\left(\frac{2\cdot 2^{-\theta^{(t)}/2H_i}}{1-2^{-1/2H_i}}\right),\end{aligned}$$

where H_i *is the maximal hitting time to node* i *(cf. Eq 2.2).*

There are two sources of "error" that our algorithm needs to control. First of all, there is a probabilistic error due to the sampling of random variables. We guarantee that this error is small with high probability by choosing $N^{(t)}$ to be large enough, formalized in Lemma 4.1.2. Second, there is a systematic bias due to truncation, such that our estimate $\hat{\pi}_i$ will be larger than π_i in expectation. Theorem 4.2.1 upper bounds this bias as a function of $\theta^{(t)}$ and the properties of the graph, i.e. the fundamental

matrix Z and the maximal hitting time H_i.

The proof of theorem 4.2.1 relies on the fact that the distribution of the return time T_i has an exponentially decaying tail (adopting the result of Aldous and Fill [1]), ensuring that the return time T_i concentrates around its mean $\mathbb{E}_i[T_i]$.

Justifying the stopping condition and bias correction. We can naively upper bound the expression $\left(Z_{ii} - \sum_{k\in\Sigma\setminus\{i\}} \mathbb{P}_i\left(X_{\theta^{(t)}} = k | \, T_i > \theta^{(t)}\right) Z_{ki}\right)$ by $\max_{j\in\Sigma} Z_{jj}$, which is a constant given the graph. This captures a notion of how quickly the graph mixes. Then the error between the estimate and the true value is upper bounded by $\frac{\mathbb{P}_i\left(T_i>\theta^{(t)}\right)}{\mathbb{E}_i[\hat{T}_i^{(t)}]}$ multiplied by this constant $\max_{j\in\Sigma} Z_{jj}$. This justifies the choice of the second stopping condition of $\frac{\hat{p}^{(t)}}{\hat{T}_i^{(t)}} < \epsilon\Delta$. Intuitively it tries to ensure that the estimate is within an $\epsilon \max_{j\in\Sigma} Z_{jj}$ multiplicative factor of the true value. In fact, for graphs that mix quickly, such as PageRank, $\max_{j\in\Sigma} Z_{jj}$ is small, so the error is likewise small. If the estimate stops at the first stopping condition, when $\frac{1}{\mathbb{E}_i[\hat{T}_i^{(t)}]} < \frac{\Delta}{1+\epsilon}$, then we have no guarantees that $\hat{p}^{(t)}$ is small. However, by Equation 4.1, with high probability $\hat{\pi}_i \geq \frac{\pi_i}{1+\epsilon}$. Therefore with high probability the estimate will have at most an additive error of $\frac{\Delta}{1+\epsilon}$ in the case that π_i is arbitrarily close to 0.

In correcting for the bias, we use an estimate of $\frac{1-\hat{p}^{(t)}}{\hat{T}_i^{(t)}}$. We adapt Theorem 4.2.1 by subtracting $\frac{\mathbb{P}_i\left(T_i>\theta^{(t)}\right)}{\mathbb{E}_i[\hat{T}_i^{(t)}]}$ to show that the expected error between the estimate $\hat{\pi}_i^C$ and the true value will be

$$\frac{\mathbb{P}_i\left(T_i > \theta^{(t)}\right)}{\mathbb{E}_i[\hat{T}_i^{(t)}]} \left(Z_{ii} - \sum_{k\in\Sigma\setminus\{i\}} \mathbb{P}_i\left(X_{\theta^{(t)}} = k | \, T_i > \theta^{(t)}\right) Z_{ki} - 1 \right).$$

Therefore, when $\max_{j\in\Sigma} Z_{jj}$ is small, subtracting by one could help to decrease the error significantly. In fact, for PageRank, we show that $\max_{j\in\Sigma} Z_{jj}$ is close to 1, thus the bias correction performs well. Of course in settings where $Z_{ii} - Z_{ki}$ is very large, then subtracting 1 through the bias correction will not significantly improve the estimate.

Convergence. The following two theorems also use the property of a exponentially decaying tail as a function of H_i to show that for large $\theta^{(t)}$, with high probability, $\mathbb{P}_i\left(T_i > \theta^{(t)}\right)$ will be small and $\mathbb{E}_i[\hat{T}_i]$ will be large (or approach $\mathbb{E}_i[T_i]$), and thus the algorithm will terminate at one of the stopping conditions.[2] The bound is a function of how sharply the distribution over return times concentrates around the mean (as analyzed in Lemma 4.2.4).

Theorem 4.2.2 states that for low probability nodes, the algorithm will terminate at stopping condition (a) for large enough iterations.

Theorem 4.2.2. *For node i such that $\pi_i < \Delta$, with probability greater than $1-\alpha$, the number of iterations is bounded above by*

$$t \leq O\left(\ln\left(H_i \ln\left(\left(\frac{1}{1-2^{-1/2H_i}}\right)\left(\frac{1}{\pi_i} - \frac{1+\epsilon}{\Delta}\right)^{-1}\right)\right)\right).$$

Theorem 4.2.3 states that for all nodes, the algorithm will terminate at stopping condition (b) for large enough iterations.

Theorem 4.2.3. *With probability greater than $1-\alpha$, the number of iterations t is bounded above by*

$$t \leq O\left(\ln\left(\frac{H_i}{\alpha}\ln\left(\pi_i\left(\frac{1}{\epsilon\Delta} + \frac{1}{1-2^{-1/2H_i}}\right)\right)\right)\right).$$

Concentration of Return Times. The following is a known lemma showing exponential tail bounds on the distribution of T_i as a function of H_i. For completeness, the proof is included here. It is an adaption of the proof presented by Aldous and Fill for continuous time Markov chains [1].

[2]This bound on may be weak because ideally, H_i should be defined as the maximal value of return times, $\mathbb{E}_j[T_i]$, only with respect to nodes j that are in the 'local' neighborhood of i; formalizing this seem to require developing framework of 'local return times, local mixing times, etc.' which is an ongoing quest in probability theory.

Lemma 4.2.4. *[1] Let $\{X_t\}$ be a finite state space Markov chain with state space Σ. For any $i \in \Sigma$ and $k \in \mathbb{Z}_+$,*

$$\mathbb{P}_i(T_i > k) \leq 2 \cdot 2^{-k/2H_i}.$$

Proof.

$$\begin{aligned}\mathbb{P}_i(T_i > m\alpha) &= \prod_{s=0}^{m-1} \mathbb{P}_i(T_i > (s+1)\alpha \mid T_i > s\alpha) \\ &= \prod_{s=0}^{m-1} \left(\sum_{j \in \Sigma} \mathbb{P}_i(T_i > (s+1)\alpha \mid X_{s\alpha} = j, T_i > s\alpha)\, \mathbb{P}_i(X_{s\alpha} = j \mid T_i > s\alpha) \right).\end{aligned}$$

By the Markov property,

$$\mathbb{P}_i(T_i > m\alpha) = \prod_{s=0}^{m-1} \left(\sum_{j \in \Sigma} \mathbb{P}_j(T_i > \alpha)\, \mathbb{P}_i(X_{s\alpha} = j \mid T_i > s\alpha) \right).$$

By Markov's inequality,

$$\begin{aligned}\mathbb{P}_i(T_i > m\alpha) &\leq \prod_{s=0}^{m-1} \left(\sum_{j \in \Sigma} \left(\frac{\mathbb{E}_j[T_j]}{\alpha} \right) \mathbb{P}_i(X_{s\alpha} = j \mid T_i > s\alpha) \right) \\ &\leq \prod_{s=0}^{m-1} \left(\frac{\max_{j \in \Sigma} \mathbb{E}_j[T_i]}{\alpha} \right) = \left(\frac{H_i}{\alpha} \right)^m.\end{aligned}$$

Substitute in $\alpha = 2H_i$ and $m = \lfloor k/2H_i \rfloor$ such that

$$\mathbb{P}_i(T_i > k) \leq \mathbb{P}_i\left(T_i > \left\lfloor \frac{k}{2H_i} \right\rfloor \cdot 2H_i \right) \leq 2^{-\lfloor k/2H_i \rfloor} \leq 2 \cdot 2^{-k/2H_i}.$$

■

Proof of Theorem 4.2.1.

The following lemma gives an exact expression of the difference between $\mathbb{E}_i[T_i]$ and $\mathbb{E}_i[\hat{T}_i^{(t)}]$. This lemma applies to both finite state space and countably infinite state space Markov chains.

Lemma 4.2.5. *For a countable state space, irreducible, aperiodic Markov chain $\{X_t\}$ with state space Σ and transition probability matrix P, and for any $i \in \Sigma$ and $t \in \mathbb{Z}_+$,*

$$\begin{aligned}\mathbb{E}_i[T_i] - \mathbb{E}_i[\hat{T}_i^{(t)}] &= \sum_{k=\theta^{(t)}}^{\infty} \mathbb{P}_i(T_i > k) \\ &= \mathbb{P}_i\left(T_i > \theta^{(t)}\right) \left(\sum_{k \in \Sigma \setminus \{i\}} \mathbb{P}_i\left(X_{\theta^{(t)}} = k \middle| T_i > \theta^{(t)}\right) \mathbb{E}_k[T_i] \right).\end{aligned}$$

Proof.

$$\begin{aligned}&\mathbb{E}_i[T_i] - \mathbb{E}_i[\hat{T}_i^{(t)}] \\ &= \sum_{k=0}^{\infty} k\mathbb{P}_i(T_i = k) - \sum_{k=0}^{\infty} \min\left(k, \theta^{(t)}\right) \mathbb{P}_i(T_i = k) \\ &= \sum_{k=0}^{\infty} \mathbb{P}_i(T_i > k) - \sum_{k=0}^{\theta^{(t)}-1} \mathbb{P}_i(T_i > k) = \sum_{k=\theta^{(t)}}^{\infty} \mathbb{P}_i(T_i > k) \\ &= \mathbb{P}_i\left(T_i > \theta^{(t)}\right) \sum_{k=\theta^{(t)}}^{\infty} \mathbb{P}_i\left(T_i > k \middle| T_i > \theta^{(t)}\right) \\ &= \mathbb{P}_i\left(T_i > \theta^{(t)}\right) \sum_{k=\theta^{(t)}}^{\infty} \sum_{q \in \Sigma \setminus \{i\}} \mathbb{P}_i\left(T_i > k \middle| X_{\theta^{(t)}} = q, T_i > \theta^{(t)}\right) \mathbb{P}_i\left(X_{\theta^{(t)}} = q \middle| T_i > \theta^{(t)}\right) \\ &= \mathbb{P}_i\left(T_i > \theta^{(t)}\right) \sum_{q \in \Sigma \setminus \{i\}} \mathbb{P}_i\left(X_{\theta^{(t)}} = q \middle| T_i > \theta^{(t)}\right) \mathbb{E}_k[T_i].\end{aligned}$$

■

We will use the exponential tail bounds to prove that the additive bias due to truncation is bounded by an expression which decays exponentially in $\theta^{(t)}$.

Lemma 4.2.6. *For a finite state space, irreducible, aperiodic Markov chain $\{X_t\}$ with state space Σ and transition probability matrix P, and for any $i \in \Sigma$ and $t \in \mathbb{Z}_+$,*

$$\frac{1}{\mathbb{E}_i[\hat{T}_i^{(t)}]} - \frac{1}{\mathbb{E}_i[T_i]} \leq \frac{\pi_i}{\mathbb{E}_i[\hat{T}]} \left(\frac{2 \cdot 2^{-\theta^{(t)}/2H_i}}{1 - 2^{-1/2H_i}} \right).$$

Proof. By Lemma 4.2.4,

$$\mathbb{E}_i[T_i] - \mathbb{E}_i[\hat{T}_i^{(t)}] \leq \sum_{k=\theta^{(t)}}^{\infty} 2 \cdot 2^{-k/2H_i} = \frac{2 \cdot 2^{-\theta^{(t)}/2H_i}}{1 - 2^{-1/2H_i}}.$$

■

To complete the proof of Theorem 4.2.1, we apply Lemma 2.1.6 to Lemma 4.2.5. This explicitly shows the error bounds as a function of the values in the fundamental matrix Z, which will be upper bounded by a constant for a given Markov chain, depending on the mixing properties of the graph.

Proof of Theorem 4.2.2.

Proof. Let t be large enough such that

$$\theta^{(t)} = 2^t \geq \frac{2H_i}{\ln 2} \ln \left(\left(\frac{2}{1 - 2^{-1/2Hi}} \right) \left(\frac{1}{\pi_i} - \frac{(1+\epsilon)}{(1-\epsilon)\Delta} \right)^{-1} \right).$$

We need $\pi_i < \frac{(1-\epsilon)\Delta}{(1+\epsilon)}$ in order for this expression to be well defined. Then,

$$\begin{aligned} \mathbb{E}_i[\hat{T}_i^{(t)}] &\geq \mathbb{E}_i[T_i] - \frac{2 \cdot 2^{-\theta^{(t)}/2H_i}}{1 - 2^{-1/2H_i}} \\ &\geq \mathbb{E}_i[T_i^{(t)}] - \left(\frac{1}{\pi_i} - \frac{(1+\epsilon)}{(1-\epsilon)\Delta} \right) \\ &= \frac{(1+\epsilon)}{(1-\epsilon)\Delta}. \end{aligned}$$

Therefore by Lemma 4.1.2, with probability greater than $1-\alpha$,

$$\begin{aligned}\hat{T}_i^{(t)} &\geq (1-\epsilon)\mathbb{E}_i[\hat{T}_i^{(t)}] \\ &\leq \frac{(1+\epsilon)}{\Delta},\end{aligned}$$

and the algorithm will stop at condition (a). Since ϵ is small, we ignore the factors $(1+\epsilon)$ and $(1-\epsilon)$. Therefore, with probability greater than $1-\alpha$, the number of iterations in the algorithm is bounded above by

$$t \leq O\left(\ln\left(H_i \ln\left(\left(\frac{}{1-2^{-1/2Hi}}\right)\left(\frac{1}{\pi_i}-\frac{1}{\Delta}\right)^{-1}\right)\right)\right).$$

■

Proof of Theorem 4.2.3.

Proof. Let t be large enough such that

$$\theta^{(t)} = 2^t \geq \frac{2H_i}{\ln 2}\ln\left(2\pi_i\left(\frac{3.5}{(1-\epsilon)\epsilon\Delta}+\frac{1}{1-2^{-1/2H_i}}\right)\right).$$

By Lemma 4.2.4,

$$\begin{aligned}\mathbb{P}_i\left(T_i > \theta^{(t)}\right) &\leq 2\cdot 2^{-\theta/2H_i} \\ &\leq \mathbb{E}_i[T_i]\left(\frac{3.5}{(1-\epsilon)\epsilon\Delta}+\frac{1}{1-2^{-1/2H_i}}\right)^{-1}.\end{aligned}$$

By Lemma 4.2.6

$$\begin{aligned}\mathbb{E}_i[\hat{T}_i^{(t)}] &\geq \mathbb{E}_i[T_i] - \frac{2\cdot 2^{-\theta^{(t)}/2H_i}}{1-2^{-1/H_i}} \\ &\geq \mathbb{E}_i[T_i] - \mathbb{E}_i[T_i]\left(\frac{3.5}{(1-\epsilon)\epsilon\Delta}+\frac{1}{1-2^{-1/2H_i}}\right)^{-1}\left(1-2^{-1/H_i}\right)^{-1}.\end{aligned}$$

And thus,

$$\begin{aligned}\frac{\mathbb{P}_i\left(T_i > \theta^{(t)}\right)}{\mathbb{E}_i[\hat{T}_i^{(t)}]} &\leq \frac{1}{\left(\frac{3.5}{(1-\epsilon)\epsilon\Delta} + \frac{1}{1-2^{-1/2H_i}}\right) - \left(1-2^{-1/2H_i}\right)^{-1}}, \\ &= \frac{1-2^{-1/2H_i}}{\left(\frac{3.5(1-2^{-1/2H_i})}{(1-\epsilon)\epsilon\Delta} + 1\right) - 1}, \\ &= \frac{(1-\epsilon)\epsilon\Delta}{3.5}.\end{aligned}$$

Therefore, with probability greater than $1-\alpha$, $\hat{T}_i^{(t)} \geq (1-\epsilon)\mathbb{E}_i[\hat{T}_i^{(t)}]$, and

$$\frac{\mathbb{P}_i\left(T_i > \theta^{(t)}\right)}{\hat{T}_i^{(t)}} \leq \frac{\mathbb{P}_i\left(T_i > \theta^{(t)}\right)}{(1-\epsilon)\mathbb{E}_i[\hat{T}_i^{(t)}]} \leq \frac{\epsilon\Delta}{3.5}.$$

We know that $N^{(t)}\hat{p}^{(t)}$ is distributed as Binomial$(N^{(t)}, p)$, where $p = \mathbb{P}_i\left(T_i > \theta^{(t)}\right)$. Therefore, we can use Chernoff's bounds (see Theorem 2.3.1) to show that if $p \geq \frac{1}{2N^{(t)}}$, then

$$\mathbb{P}\left(\hat{p}^{(t)} \leq 3.5p\right) \geq 1 - e^{\frac{-2.5^2 N^{(t)} p}{3}} \geq 1 - e^{-6.25/6} \geq \frac{1}{2}.$$

If $p \leq \frac{1}{2N^{(t)}}$, then

$$\mathbb{P}(\hat{p}^{(t)} == 0) = (1-p)^{N^{(t)}} > 1 - pN^{(t)} > \frac{1}{2}.$$

Therefore, with probability greater than $\frac{1}{2}$, the algorithm will terminate due to stopping condition (b). If the algorithm does not terminate in this iteration, then the algorithm will terminate in the next iteration with probability greater than $\frac{1}{2}$. Thus after another additional $\log_2(1/\alpha)$ iterations, we can guarantee that with probability greater than $1-\alpha$, the algorithm will have already terminated by then.

Therefore, since $\theta^{(t)} = 2^t$, with probability greater than $1-\alpha$, the number of

iterations needed in the algorithm is upper bounded by

$$t \leq \log_2(1/\alpha) + \log_2\left(\frac{2H_i}{\ln 2}\ln\left(2\pi_i\left(\frac{3.5}{(1-\epsilon)\epsilon\Delta} + \frac{1}{1-2^{-1/2H_i}}\right)\right)\right)$$
$$= \log_2\left(\frac{2H_i}{\alpha \ln 2}\ln\left(2\pi_i\left(\frac{3.5}{(1-\epsilon)\epsilon\Delta} + \frac{1}{1-2^{-1/2H_i}}\right)\right)\right)$$

■

■ 4.3 Countable-state space Markov Chain

The proof of Theorems 4.2.1, 4.2.2, and 4.2.3 required the state space of the Markov chain to be finite, so that we could upper bound the tail of the distribution of T_i using the maximal hitting time H_i. In fact, these results can be extended to many countably infinite state space Markov chains as well. We prove that the tail of the distribution of T_i decays exponentially for any node i in a countable state space Markov chain that satisfies Assumption 4.3.1.

Assumption 4.3.1. *The Markov chain $\{X_t\}$ is aperiodic and irreducible. There exists a Lyapunov function $V : \Sigma \to \mathbb{R}_+$ and constants $\nu_{\max}, \gamma > 0$, and $b \geq 0$, that satisfy the following conditions:*

1. *The set $B = \{x \in \Sigma : V(x) \leq b\}$ is finite,*

2. *For all $x, y \in \Sigma$ such that $\mathbb{P}(X_{t+1} = j | X_t = i) > 0$,*

$$|V(j) - V(i)| \leq \nu_{\max},$$

3. *For all $x \in \Sigma$ such that $V(x) > b$,*

$$\mathbb{E}[V(X_{t+1}) - V(X_t) | X_t = x] < -\gamma.$$

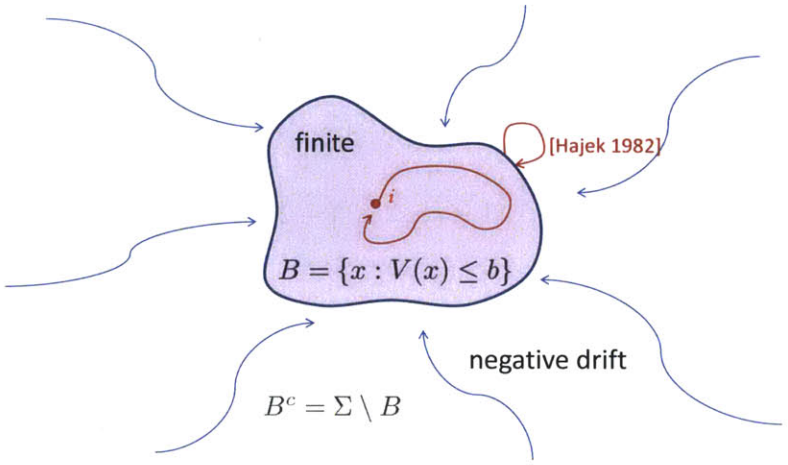

Figure 4.1. Lyapunov decomposition of SS

At first glance, this may seem very restrictive. But in fact, this is quite reasonable: by the Foster-Lyapunov criteria (see Theorem 2.2.1), a countable state space Markov chain is positive recurrent if and only if there exists a Lyapunov function $V : \Sigma \to \mathbb{R}_+$ that satisfies condition (1) and (3), (2'): $\mathbb{E}[V(X_{t+1})|X_t = x] < \infty$ for all $x \in \Sigma$. That is, Assumption 4.3.1 has (2), which is a restriction of the condition (2'). The implications of Assumption 4.3.1 can be visualized in Figure 4.1. The existence of the Lyapunov function allows us to decompose the state space into sets B and B^c such that for all nodes $x \in B^c$, there is an expected decrease in the Lyapunov function in the next step or transition. Therefore, for all nodes in B^c there is a negative drift towards set B. In addition, in any single step the random walk cannot escape "too far".

Using the concentration bounds for the countable state space settings, we can prove the following theorems that parallel the theorems stated for the finite state space setting. The formal statements are restricted to nodes in $B = \{i \in \Sigma : V(i) \leq b\}$. This is not actually restrictive as for any i such that $V(i) > b$, we can define a new Lyapunov function where $V'(i) = b$, and $V'(j) = V(j)$ for all $j \neq i$. Then $B' = B \cup \{i\}$, and V' still satisfies assumption 4.3.1 for new values of $\nu_{\max}$, γ, and b. Therefore, by no means

is our result restrictive.

Theorem 4.3.2. *For a Markov chain satisfying Assumption 4.3.1, and for any $i \in \Sigma$,*

$$\frac{1}{\mathbb{E}_i[\hat{T}_i^{(t)}]} - \pi_i = \frac{\mathbb{P}_i\left(T_i > \theta^{(t)}\right)}{\mathbb{E}_i[\hat{T}_i^{(t)}]} \left(\sum_{k \in \Sigma \setminus \{i\}} \mathbb{P}_i\left(X_{\theta^{(t)}} = k | \, T_i > \theta^{(t)}\right) \frac{\mathbb{E}_k[T_i]}{\mathbb{E}_i[T_i]} \right)$$
$$\leq \frac{\pi_i}{\mathbb{E}_i[\hat{T}^{(t)}]} \left(\frac{4 \cdot 2^{-\theta^{(t)}/R_i}}{1 - 2^{-1/R_i}} \right)$$

where R_i is defined such that

$$R_i = O\left(\frac{H_i^B e^{2\eta\nu_{\max}}}{(1-\rho)(e^{\eta\nu_{\max}} - \rho)} \right).$$

H_i^B is the maximal hitting time over the Markov chain with its state space restricted to the subset B. The scalars η and ρ are functions of γ and $\nu_{\max}$ (see Eq 2.5).

The following two theorems give bound on the convergence of the algorithm.

Theorem 4.3.3. *For node i such that $\pi_i < \Delta$, with probability greater than $1 - \alpha$, the number of iterations is bounded above by*

$$t \leq O\left(\ln\left(R_i \ln\left(\left(\frac{1}{1 - 2^{-1/R_i}} \right) \left(\frac{1}{\pi_i} - \frac{1+\epsilon}{\Delta} \right)^{-1} \right) \right) \right).$$

Theorem 4.3.4. *With probability greater than $1 - \alpha$, the number of iterations t is bounded above by*

$$t \leq O\left(\ln\left(\frac{R_i}{\alpha} \ln\left(\pi_i \left(\frac{1}{\epsilon\Delta} + \frac{1}{1 - 2^{-1/R_i}} \right) \right) \right) \right).$$

In order to prove these lemma, we need to prove a statement equivalent to Lemma 4.2.4. Building upon results of [17], we extend the result of Aldous and Fill [1], we establish that indeed return times have exponentially decaying tail even for countable-

state space Markov chain as long as they satisfy Assumption 4.3.1. First, we introduce some notations that will be useful to state and prove these results.

Useful notation. We introduce formal notation for observing $\{X_t\}_{t\geq 0}$ over the subset B. Let $\{Y_t\}_{t\geq 0}$ be a Markov chain with B as its state space. $\{Y_t\}$ is a subsequence of $\{X_t\}$ constructed in the following way. Define the subsequence $\{S_k\}$ of $\mathbb{Z}_+$ as:

$$S_0 = 0, S_k = \min\{t > S_{k-1} : X_t \in B\},$$

and define $Y_t = X_{S_t}$ such that

$$P_Y(x, y) = \mathbb{P}\left(X_{S_1} = y | X_0 = x\right) \quad \text{for any } x, y \in B.$$

Let $Q_t = S_{t+1} - S_t$, the length of the path between X_{S_t} and $X_{S_{t+1}}$.

Let $T_i^B = \inf\{t \geq 1 \mid Y_t = i\}$, the return time to i for the chain $\{Y_t\}$.

Let $H_i^B = \max_{j\in B} \mathbb{E}_j\left[T_i^B\right]$, the maximal expected hitting time to node i for the chain $\{Y_t\}$. We can use these variables to express the return time to node i by

$$T_i = S_{T_i^B} = \sum_{k=0}^{T_i^B - 1} Q_k. \tag{4.2}$$

Lemma 4.3.5. *Let $\{X_t\}$ be a countable state space Markov chain satisfying Assumption 4.3.1. Let $\{Y_t\}$ be defined above as the Markov chain restricted to B, and let Q_k be the length between visits to B. For any $W \in \mathbb{Z}_+$,*

$$\mathbb{P}_i\left(\sum_{k=0}^{W-1} Q_k > Z\right) \leq \exp\left(\frac{0.8(1-\rho)}{e^{\eta\nu_{\max}}}\left(\frac{1.25We^{2\eta\nu_{\max}}}{(1-\rho)(e^{\eta\nu_{\max}} - \rho)} + W - Z\right)\right).$$

The constants γ and $\nu_{\max}$ are given by the Assumption 4.3.1, and the scalars η and ρ are functions of γ and $\nu_{\max}$ (Eq. 2.5).

Proof. By the law of iterated expectation,

$$\mathbb{E}_i\left[\prod_{k=0}^{W-1}\exp\left(\lambda Q_k\right)\right]=\mathbb{E}_i\left[\exp\left(\lambda Q_0\right)\mathbb{E}_i\left[\prod_{k=1}^{W-1}\exp\left(\lambda Q_k\right)\middle| Q_0\right]\right]$$

Conditioned on $Y_1 = j$, Y_0 and Q_0 are independent of $\{Q_k\}_{k>0}$, because $Y_1 = X_{Q_0}$. Thus by the strong Markov property,

$$\mathbb{E}_i\left[\prod_{k=1}^{W-1}\exp\left(\lambda Q_k\right)\middle| Q_0\right]\leq\max_{j\in B}\mathbb{E}\left[\prod_{k=1}^{W-1}\exp\left(\lambda Q_k\right)\middle| Y_1=j\right],$$

so that

$$\mathbb{E}_i\left[\prod_{k=0}^{W-1}\exp\left(\lambda Q_k\right)\right]\leq\mathbb{E}_i[\exp(\lambda Q_0)]\max_{j\in B}\mathbb{E}\left[\prod_{k=1}^{W-1}\exp\left(\lambda Q_k\right)\middle| Y_1=j\right].$$

We iteratively apply conditioning to show that

$$\mathbb{E}_i\left[\prod_{k=0}^{W-1}\exp\left(\lambda Q_k\right)\right]\leq\mathbb{E}_i[\exp(\lambda Q_0)]\prod_{k=1}^{W-1}\left(\max_{j\in B}\mathbb{E}\left[\exp\left(\lambda Q_k\right)|\, Y_k=j\right]\right).$$

We can upper bound Q_k by assuming that it always goes on an excursion from B, such that

$$Q_k\leq 1+(\text{length of an excursion into } B^c).$$

We invoke Hajek's result of Theorem 2.2.2, with $V(x) < b + \nu_{\max}$ to bound the exponential moments of the excursion,

$$\mathbb{E}_i\left[\prod_{k=0}^{W-1}\exp\left(\lambda Q_k\right)\right]\leq\left(e^{\lambda}\left(e^{\eta\nu_{\max}}\left(\frac{e^{\lambda}-1}{1-\rho e^{\lambda}}\right)+1\right)\right)^{W}.$$

For $\lambda < \min\left(0.43, \frac{0.8(1-\rho)}{\rho}\right)$,

$$e^{\lambda} < 1 + 1.25\lambda \quad \text{and} \quad 1 - \rho e^{\lambda} > 1 - \rho - 1.25\rho\lambda.$$

By substituting in these approximations and using $1 + x < e^x$, we obtain

$$\begin{aligned}\left(e^{\lambda}\left(e^{\eta\nu_{\max}}\left(\frac{e^{\lambda}-1}{1-\rho e^{\lambda}}\right)+1\right)\right)^{W} &< \left(e^{\lambda}\left(e^{\eta\nu_{\max}}\left(\frac{1.25\lambda}{1-\rho-1.25\rho\lambda}\right)+1\right)\right)^{W} \\ &< \exp(\lambda W)\exp\left(\frac{1.25\lambda W e^{\eta\nu_{\max}}}{1-\rho-1.25\rho\lambda}\right) \\ &< \exp\left(\lambda W\left(\frac{1.25e^{\eta\nu_{\max}}}{1-\rho-1.25\rho\lambda}+1\right)\right).\end{aligned}$$

By Markov's inequality,

$$\begin{aligned}\mathbb{P}_i\left(\sum_{k=0}^{W-1} Q_k > Z\right) &\leq \frac{\mathbb{E}_i\left[\exp\left(\lambda\sum_{k=0}^{W-1} Q_k\right)\right]}{\exp(\lambda Z)} \\ &\leq \exp\left(\lambda W\left(\frac{1.25e^{\eta\nu_{\max}}}{1-\rho-1.25\rho\lambda}+1\right)-\lambda Z\right).\end{aligned}$$

Choose $\lambda = \frac{0.8(1-\rho)}{e^{\eta\nu_{\max}}}$. We can easily check using Eq 2.5 that $\lambda < \max\left(0.43, \frac{0.8(1-\rho)}{\rho}\right)$ always holds. Therefore, we complete the proof by substituting in for λ,

$$\mathbb{P}_i\left(\sum_{k=0}^{W-1} Q_k > Z\right) \leq \exp\left(\frac{0.8(1-\rho)}{e^{\eta\nu_{\max}}}\left(\frac{1.25We^{2\eta\nu_{\max}}}{(1-\rho)(e^{\eta\nu_{\max}}-\rho)}+W-Z\right)\right).$$

■

Next we prove that the distribution of the return time T_i over the countably infinite state space Markov chain is bounded by a decaying exponential function. This result is the equivalent to Lemma 4.2.4 for countably infinite state space Markov chains.

Theorem 4.3.6. *Let $\{X_t\}$ be an irreducible, aperiodic Markov chain satisfying As-*

sumption 4.3.1. For any $i \in B$ and for all $k \in \mathbb{Z}_+$,

$$\mathbb{P}_i\left(T_i > k\right) \leq 4 \cdot 2^{-\frac{k}{R_i}},$$

where

$$R_i = 2H_i^B\left(\frac{1.25e^{2\eta\nu_{\max}}}{(1-\rho)(e^{\eta\nu_{\max}}-\rho)}+1\right)+\frac{\ln(2)e^{\eta\nu_{\max}}}{0.8(1-\rho)}.$$

Proof. By Eq. 4.2, for any constants $W, Z \in \mathbb{Z}_+$,

$$\{T_i^B \leq W\} \bigcap \left\{\sum_{k=0}^{W-1} Q_k \leq Z\right\} \implies \{T_i \leq Z\}.$$

We use the union bound on the contrapositive statement to obtain the inequalities

$$\begin{aligned}\mathbb{P}_i(T_i > Z) &\leq \mathbb{P}_i\left(\{T_i^B > W\} \cup \left\{\sum_{k=0}^{W-1} Q_k > Z\right\}\right) \\ &\leq \mathbb{P}_i(T_i^B > W) + \mathbb{P}_i\left(\sum_{k=0}^{W-1} Q_k > Z\right). \end{aligned} \tag{4.3}$$

Choose $W = 2H_i^B\left(2+\frac{k}{R_i}\right)$ and

$$Z = 4H_i^B\left(\frac{1.25e^{2\eta\nu_{\max}}}{(1-\rho)(e^{\eta\nu_{\max}}-\rho)}+1\right)+\frac{\ln(2)e^{\eta\nu_{\max}}}{0.8(1-\rho)}+k = 2R_i - \frac{\ln(2)e^{\eta\nu_{\max}}}{0.8(1-\rho)}+k.$$

The next inequality follows from substituting these expressions for W and Z into equation 4.3, and applying Lemmas 4.2.4 and 4.3.5:

$$\mathbb{P}_i\left(T_i > 2R_i - \frac{\ln(2)e^{\eta\nu_{\max}}}{0.8(1-\rho)}+k\right) \leq 2^{-\frac{k}{R_i}},$$

$$\mathbb{P}_i\left(T_i > 2R_i + k\right) \leq \mathbb{P}_i\left(T_i > 2R_i - \frac{\ln(2)e^{\eta\nu_{\max}}}{0.8(1-\rho)}+k\right) \leq 2^{-\frac{k}{R_i}},$$

$$\mathbb{P}_i\left(T_i > k\right) \leq 2^{-\frac{k-2R_i}{R_i}} \leq 4 \cdot 2^{-\frac{k}{R_i}}.$$

■

The remaining steps for the proofs of Theorems 4.3.2, 4.3.3, and 4.3.4 simple follow the proofs for the corresponding Theorems 4.2.1, 4.2.2, and 4.2.3, by replacing uses of Lemma 4.2.4 with Lemma 4.3.6.

Chapter 5

Estimating Multiple Nodes Simultaneously

■ 5.1 Multiple Node (MCRT-Multi) Algorithm

The algorithm we presented in Chapter 3 estimates the stationary probability of a single node. The algorithm is especially relevant in the setting when there is only a small set of nodes of interest. In this section, we introduce a modification to the algorithm that allows us to obtain estimates for all the nodes simultaneously. We still fix a target node i, ideally a node with high importance and thus smaller computation time. In addition to storing only the length of the sampled random walks, we also store the number of visits to other nodes in the network. If we are interested in estimating the stationary probabilities of a subset of nodes $A \subset \Sigma$, then the algorithm requires up to $|A|$ space to keep track of the number of visits to each of these nodes. The modified algorithm is presented below.

$\{X_t\}_{t\geq 0}$ is a Markov chain on state space Σ having transition probability matrix $P : \Sigma \times \Sigma \to [0, 1]$. Let $X_0 = i$, such that the random walk always begins at node i. Define F_j to be the total number of visits to node j before time T_i.

$$F_j = \sum_{t=1}^{\infty} \mathbf{1}_{\{X_t=j\}} \mathbf{1}_{\{t\leq T_i\}} = \sum_{t=1}^{T_i} \mathbf{1}_{\{X_t=j\}}$$

By property of positive recurrent Markov chains, for all $j \in \Sigma$ (see Eq. 2.4),

$$\pi_j = \frac{\mathbb{E}_i[F_j]}{\mathbb{E}_i[T_i]} = \frac{\mathbb{E}_i[\text{visits to } j \text{ along sample path}]}{\mathbb{E}_i[\text{length of sample path}]}$$

Below is the formal description of the algorithm. The only modification is effectively in Step 1, where we keep track of the frequency of visits to nodes in addition to the length of the random walk as before.

MCRT-Multi Algorithm

Initialize Choose parameters Δ = threshold for importance, ϵ = closeness of the estimate, and α = probability of failure. Set

$$t = 1, \theta^{(1)} = 2, N^{(1)} = \left\lceil \frac{6(1+\epsilon)\ln(8/\alpha)}{\epsilon^2} \right\rceil.$$

Step 1 (Gather Samples) For each k in $\{1, 2, 3, \ldots N^{(t)}\}$, generate sample paths of the Markov chain beginning at node i, denoted by $\{X_0^{(k)}, X_1^{(k)}, X_2^{(k)}, \ldots\}$ where $X_0^{(k)} = i$ and $\mathbb{P}\left(X_{m+1}^{(k)} = j | X_m^{(k)} = h\right) = P_{hj}$. Recall from Eq. 2.1 that $T_i = \inf\{m \geq 1 \mid X_m = i\}$. Let

$$s_k = \min(T_i, \theta),$$

and

$$f_k(j) = \sum_{m=1}^{\infty} \mathbf{1}_{\{X_m = j\}} \mathbf{1}_{\{m \leq s_k\}} = \sum_{m=1}^{s_k} \mathbf{1}_{\{X_m = j\}}.$$

Let $p^{(t)}$ = fraction of samples that were truncated at $\theta^{(t)}$, and let

$$\hat{F}_j^{(t)} = \frac{1}{N^{(t)}} \sum_{k=1}^{N^{(t)}} f_k(j),$$

and

$$\hat{T}_i^{(t)} = \frac{1}{N^{(t)}} \sum_{k=1}^{N^{(t)}} s_k.$$

Step 2 (Termination Conditions) If either

$$\textbf{(a)} \quad \hat{T}_i^{(t)} > \frac{(1+\epsilon)}{\Delta} \qquad \text{or} \qquad \textbf{(b)} \quad \frac{p^{(t)}}{\hat{T}_i^{(t)}} < \epsilon\Delta,$$

then stop and return estimate

$$\hat{\pi}_j = \frac{\hat{F}_j^{(t)}}{\hat{T}_i^{(t)}}, \qquad \forall j \in \Sigma.$$

Step 3 (Update Rules) Set

$$\theta^{(t+1)} \leftarrow 2 \cdot \theta^{(t)}, N^{(t+1)} \leftarrow \left\lceil \frac{3(1+\epsilon)\theta^{(t+1)} \ln(4\theta^{(t+1)}/\alpha)}{\hat{T}_i^{(t)} \epsilon^2} \right\rceil, \text{ and } t \leftarrow t+1.$$

Return to **Step 1**.

Note that $\hat{\pi}_i = \frac{1-p^{(t)}}{\hat{T}_i^{(t)}}$, i.e. the same as the estimate in the original algorithm with the bias correction.

■ 5.2 Theoretical Analysis

This algorithm requires the same number of steps and neighbor queries to the network, thus the time complexity analysis is the same. Because $\hat{F}_j^{(t)}$ is the sum of bounded independent identically distributed random variables, it also concentrates around $\mathbb{E}_i[\hat{F}_j^{(t)}]$ for large N. The main analysis required is to argue that $\mathbb{E}_i[\hat{F}_j^{(t)}]$ is close to $\mathbb{E}_i[F_j]$. Of course as $\theta^{(t)}$ goes to infinity, and $\mathbb{E}_i[\hat{F}_j^{(t)}]$ approaches $\mathbb{E}_i[F_j]$. The result is stated for finite state space Markov chains.

Theorem 5.2.1. *For a finite state space, irreducible, aperiodic Markov chain $\{X_t\}$ with state space Σ and transition probability matrix P, and for any $i, j \in \Sigma$,*

$$\frac{\mathbb{E}_i[\hat{F}_j^{(t)}]}{\mathbb{E}_i[\hat{T}_i^{(t)}]} - \pi_j = \frac{\mathbb{P}\left(T_i > \theta^{(t)}\right)}{\mathbb{E}_i[\hat{T}_i^{(t)}]} \left(Z_{ij} - \sum_{k \in \Sigma \setminus \{i\}} \mathbb{P}_i\left(X_{\theta^{(t)}} = k \middle| T_i > \theta^{(t)} \right) Z_{kj} \right)$$

Proof. By definition,

$$\begin{aligned}
&\mathbb{E}_i[F_j] - \mathbb{E}_i[\hat{F}_j^{(t)}] \qquad (5.1)\\
&= \mathbb{E}_i\left[\sum_{t=1}^{\infty} \mathbf{1}_{\{X_t = j\}} \mathbf{1}_{\{t \le T_i\}}\right] - \mathbb{E}_i\left[\sum_{t=1}^{\infty} \mathbf{1}_{\{X_t = j\}} \mathbf{1}_{\{t \le T_i\}} \mathbf{1}_{\{t \le \theta^{(t)}\}}\right]\\
&= \mathbb{P}\left(T_i > \theta^{(t)}\right) \mathbb{E}_i\left[\sum_{t=1}^{\infty} \mathbf{1}_{\{X_t = j\}} \mathbf{1}_{\{t \le T_i\}} \middle| T_i > \theta^{(t)}\right]\\
&= \mathbb{P}\left(T_i > \theta^{(t)}\right) \sum_{k \in \Sigma \setminus \{i\}} \mathbb{P}_i\left(X_{\theta^{(t)}} = k \middle| T_i > \theta^{(t)} \right) \mathbb{E}_i\left[\sum_{t=\theta^{(t)}+1}^{\infty} \mathbf{1}_{\{X_t = j\}} \mathbf{1}_{\{t \le T_i\}} \middle| X_{\theta^{(t)}} = k\right]\\
&= \mathbb{P}\left(T_i > \theta^{(t)}\right) \sum_{k \in \Sigma \setminus \{i\}} \mathbb{P}_i\left(X_{\theta^{(t)}} = k \middle| T_i > \theta^{(t)} \right) \mathbb{E}_k\left[\sum_{t=1}^{\infty} \mathbf{1}_{\{X_t = j\}} \mathbf{1}_{\{t \le T_i\}}\right]. \qquad (5.2)
\end{aligned}$$

By Lemma 2.1.7,

$$\begin{aligned}
&\mathbb{E}_i[F_j] - \mathbb{E}_i[\hat{F}_j^{(t)}]\\
&= \mathbb{P}\left(T_i > \theta^{(t)}\right) \sum_{k \in \Sigma \setminus \{i\}} \mathbb{P}_i\left(X_{\theta^{(t)}} = k \middle| T_i > \theta^{(t)} \right) \pi_j (E_k[T_i] + E_i[T_j] - E_k[T_j]).
\end{aligned}$$

For $i = j$, we have a simpler expression.

$$\mathbb{E}_i[F_i] - \mathbb{E}_i[\hat{F}_i^{(t)}] = \mathbb{P}\left(T_i > \theta^{(t)}\right).$$

We combine equations 5.2 with Lemma 4.2.5 to show that

$$\frac{\mathbb{E}_i[\hat{F}_j^{(t)}]}{\mathbb{E}_i[\hat{T}_i^{(t)}]} - \pi_j = \frac{\mathbb{E}_i[\hat{F}_j^{(t)}]\mathbb{E}_i[T_i] - \mathbb{E}_i[F_j]\mathbb{E}_i[\hat{T}_i^{(t)}]}{\mathbb{E}_i[T_i]\mathbb{E}_i[\hat{T}_i^{(t)}]}$$

$$\frac{\mathbb{E}_i[\hat{F}_j^{(t)}]}{\mathbb{E}_i[\hat{T}_i^{(t)}]} - \pi_j = \frac{\mathbb{E}_i[F_j]}{\mathbb{E}_i[\hat{T}_i^{(t)}]}\left(\frac{\mathbb{E}_i[\hat{F}_j^{(t)}]}{\mathbb{E}_i[F_j]} - \frac{\mathbb{E}_i[\hat{T}_i^{(t)}]}{\mathbb{E}_i[T_i]}\right)$$

$$= \frac{\mathbb{E}_i[F_j]}{\mathbb{E}_i[\hat{T}_i^{(t)}]}\left(\frac{\mathbb{E}_i[\hat{F}_j^{(t)}] - \mathbb{E}_i[F_i]}{\mathbb{E}_i[F_i]} - \frac{\mathbb{E}_i[\hat{T}_i^{(t)}] - \mathbb{E}_i[T_i]}{\mathbb{E}_i[T_i]}\right)$$

$$= \frac{\pi_j \mathbb{P}\left(T_i > \theta^{(t)}\right)}{\mathbb{E}_i[\hat{T}_i^{(t)}]} \sum_{k \in \Sigma \setminus \{i\}} \mathbb{P}_i\left(X_{\theta^{(t)}} = k \middle| T_i > \theta^{(t)}\right)(E_i[T_j] - E_k[T_j]).$$

By Lemma 2.1.6,

$$\frac{\mathbb{E}_i[\hat{F}_j^{(t)}]}{\mathbb{E}_i[\hat{T}_i]} - \pi_j = \frac{\mathbb{P}\left(T_i > \theta^{(t)}\right)}{\mathbb{E}_i[\hat{T}_i^{(t)}]}\left(\sum_{k \in \Sigma \setminus \{i\}} \mathbb{P}_i\left(X_{\theta^{(t)}} = k \middle| T_i > \theta^{(t)}\right) Z_{kj} - Z_{ij}\right).$$

■

Notice the similarity between the expressions in Theorem 4.2.1 and Theorem 5.2.1. Again the error is small when $\frac{\mathbb{P}_i(T_i > \theta^{(t)})}{\mathbb{E}_i[\hat{T}_i]}$ is small and when the graph mixes quickly, i.e. Z_{ij} is not too large. Thus the error is a function of the properties of both node i and node j. Remember that $\mathbb{P}_i(T_i > \theta^{(t)})$ decays exponentially in $\theta^{(t)}$. Z_{kj} is a fixed constant for a fixed Markov chain, thus the expression within the parentheses is upper bounded by a constant. This expression can be either positive or negative, thus we cannot guarantee that the estimate will be an upper or lower bound. It shows how the error is a function of the fundamental matrix Z (relating to the mixing time). We will show in following simulations that for graphs that mix quickly (such as PageRank), Z_{ij} is small for all $i, j \in \Sigma$, and therefore this algorithm performs very well.

Chapter 6

Examples and Simulations

We discuss a few applications of our algorithm to concrete examples of Markov chains. The examples illustrate the wide applicability of our local algorithm for estimating stationary distribution.

■ 6.1 Queueing System

In queuing theory, Markov chains are often used to model the size of the queue at a server, which evolves over time as requests arrive and are processed. For illustrative purposes, we chose the MM1 queue, equivalent to a random walk on $\mathbb{Z}_+$. Assume we have a single server where the requests arrive according to a Poisson process, and the processing time for a single request is distributed exponentially. We can model the queue length with the random walk shown in Figure 6.1, where $p = \mathbb{P}$(a new request arrives before current request is fully processed).

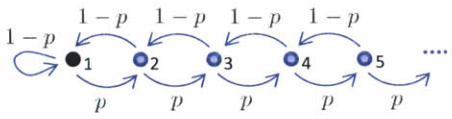

Figure 6.1. MM1 Queue.

Algorithm results. In Figures 6.2, 6.3, and 6.4, we show the results of applying our algorithm to estimate the stationary distribution of the first 50 nodes in the MM1

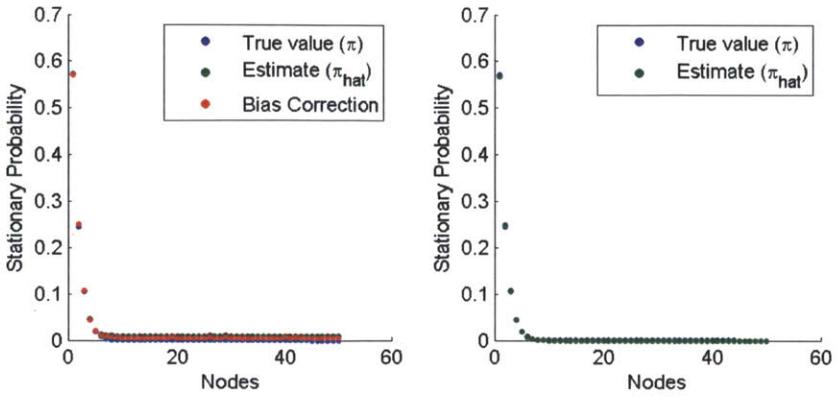

Figure 6.2. MM1 Queue —(left) Estimates from the MCRT algorithm, (right) Estimates from the MCRT-Multi algorithm.

queue model. For Figures 6.2 and 6.3, we choose parameters $\Delta = 0.02$, $\epsilon = 0.15$, and $\alpha = 0.2$. The graph on the left of Figure 6.2 shows the results of running the MCRT Algorithm for every single node $i \in \{1, 2, 3, \ldots, 50\}$ separately. The graph on the right of Figure 6.2 shows the results of running the MCRT-Multi Algorithm for i chosen to be node 1 (the leftmost node). Figure 6.3 shows the final values for N, θ, and $\hat{p}$ for the execution of algorithm MCRT on each node $i \in \{1, 2, 3 \ldots, 50\}$. The stationary probabilities decay exponentially, thus only the first few nodes are of importance.

These figures show that all variants of our algorithm perform extremely well. From Figure 6.3, we can observe the behavior of the algorithm as a function of the nodes. As i increases, π_i decreases, and the computation required increases as expected. However, for $\pi_i \ll \Delta$, the algorithm will terminate early and thus the computation cost levels off to a constant. This shows how the truncation helps to save computation for nodes with small π_i.

Algorithm dependence on Δ. Figure 6.4 shows the results of our algorithm as a function of Δ. The plot shows the results from three different nodes, chosen to illustrate the behavior on nodes with varying π_i. The graph on the left plots the estimates for the stationary probability and the graph on the right plots the total number of steps

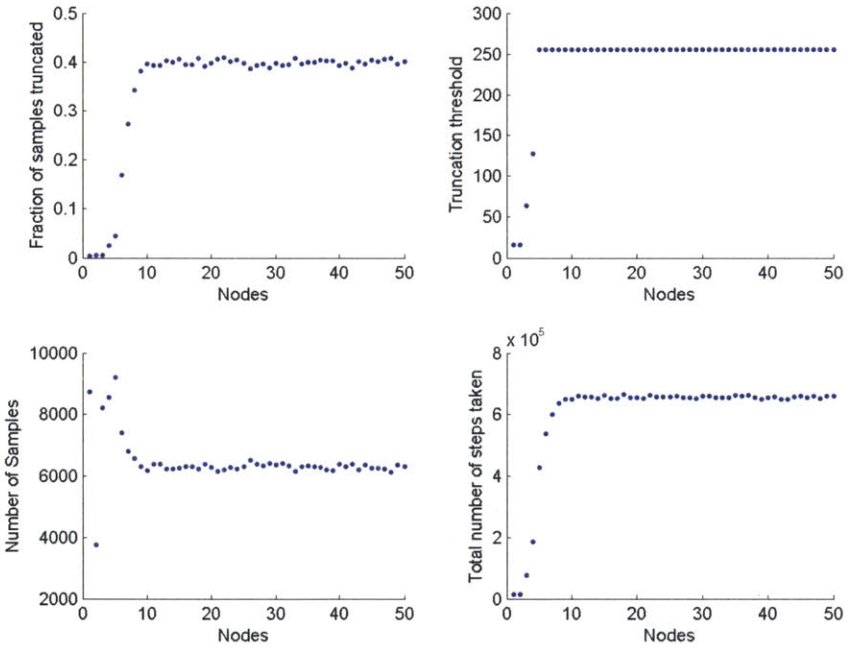

Figure 6.3. MM1 Queue —Statistics from the MCRT algorithm

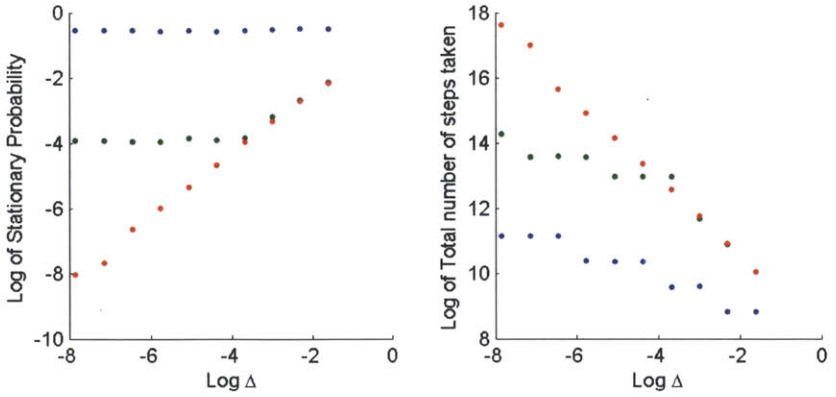

Figure 6.4. MM1 Queue —Results of the MCRT algorithm as a function of Δ

taken by the algorithm. The figures are shown on a log-log scale. We can observe that the computation time of the algorithm behaves as $\frac{1}{\Delta}$. The blue node has the largest stationary probability, therefore the algorithm will terminate at stopping condition (b) for small $\hat{p}$. Thus, as Δ decreases, the stationary probability estimate remains the same, and the computation time increases only slowly in $\frac{1}{\Delta}$. The red node has the smallest stationary probability, $\pi_i \ll \Delta$. Therefore, the algorithm terminates at stopping condition (a), which is verified by observing that the estimate $\hat{\pi}_i$ behaves as Δ. For similar reasons, the computation behaves as $\frac{1}{\Delta}$. The green node shows both behaviors. When $\Delta > \pi_i$, then $\hat{\pi}_i$ behaves as Δ, and the computation time behaves as $\frac{1}{\Delta}$. When $\Delta < \pi_i$, then $\hat{\pi}_i$ concentrates around π_i and no longer decreases with Δ. The computation time levels off and grows slowly.

Tight characterization of return times - Comparison with theoretical bounds. In fact, due to the simplicity of this random walk, we can show analytical expressions for the distribution over return times to the node 1. Given the distribution over $\hat{T}_i^{(t)}$, we can show tighter bounds on the running time of the algorithm for this Markov chain. We then compare this to the bounds obtained by using the looser but more general characterization for concentration of T_i in Lemma 4.3.6. We verify that the two bounds

exhibit the same qualitative behavior.

Let the node of interest i be the leftmost node. The explicit expression for the distribution of the return times is given by counting paths that return to i. Since there are no self loops, the path can be seen as nested pairs of transitions such that for every transition j to $j+1$, there must be a matching transition $j+1$ to j. Therefore, we can count paths using the Catalan numbers (i.e. number of possible arrangements of pairs of parentheses).

$$\mathbb{P}_i(T_i = k) = \begin{cases} 1-p & \text{if } k = 1 \\ 0 & \text{if } k \text{ is odd and } > 1 \\ \left(\frac{(2m-2)!}{m!(m-1)!}\right) p^m (1-p)^m & \text{if } k = 2m. \end{cases}$$

In order to obtain a more understandable expression for $\mathbb{P}_i(T_i > k)$, we use Stirling's approximation. Then we approximate the sum by an integral, and use integration by parts to get an upper bound. We can show that this expression is tight for large k.

$$\mathbb{P}_i(T_i > k) \leq -\frac{1}{\ln(4p(1-p))} \left(\frac{(4p(1-p))^{k+2}}{2\pi(k+2)^3}\right)^{1/2} \tag{6.1}$$

We use similar techniques and lemma 4.2.5 to get an expression for $\mathbb{E}_i[\min(T_i, \theta)]$, which is again tight for large θ.

$$\mathbb{E}_i[\min(T_i, \theta)] \geq \frac{1-p}{1-2p} - \frac{2}{(\ln(4p(1-p)))^2} \left(\frac{(4p(1-p))^{\theta+2}}{2\pi(\theta+2)^3}\right)^{1/2} \tag{6.2}$$

Using Lyapunov function analysus, we can obtain more generic bounds. Verify that the linear function $V(x) = x$ indeed satisfies assumption 4.3.1, with $\nu_{\max} = 1$ and

$\gamma = 1 - 2p$. By equation 2.5, $\eta = \frac{1-2p}{2(e-2)}$ and $\rho = 1 - \frac{(1-2p)^2}{4(e-2)}$. We directly apply Hajek's result in Theorem 2.2.2 to obtain the following bounds:

$$\mathbb{P}_i(T_i > k) \leq pe^{\eta}\rho^k$$

$$\mathbb{E}_i[\min(T_i, \theta)] \geq \frac{1-p}{1-2p} - pe^{\eta}\frac{\rho^{\theta}}{1-\rho}.$$

We can use these bounds to compute for the algorithm runtimes. Recall the two stopping conditions of the algorithm. Figure 6.5 show the minimum θ such that either of the following inequalities hold:

$$\mathbb{E}_i[\min(T_i, \theta)] = \mathbb{E}_i[T_i] - \sum_{k=\theta}^{\infty} \mathbb{P}_i(T_i > k) > \frac{1+\epsilon}{\Delta},$$

$$\frac{\mathbb{P}_i(T_i > \theta)}{\mathbb{E}_i[\min(T_i, \theta)]} \leq \epsilon\Delta.$$

The Figure shows values for θ such that with probability greater than $(1 - \alpha)$, the algorithm will have terminated by the time $\theta^{(t)} \geq \theta$. The figure plots θ as a function of p, or the amount of "drift", and α, governing the probabilistic guarantee. As expected, the bound by using the generic Lyapunov function technique, denoted by curve (a), is looser than the bound obtained by using an exact characterization, denoted by curve (b). We can verify that the two bounds have similar qualitative behavior. As expected, when p increases, the drift back towards node 1 is smaller, thus the algorithm requires larger θ. When α decreases, the algorithm requires larger θ to guarantee termination with probability $1 - \alpha$.

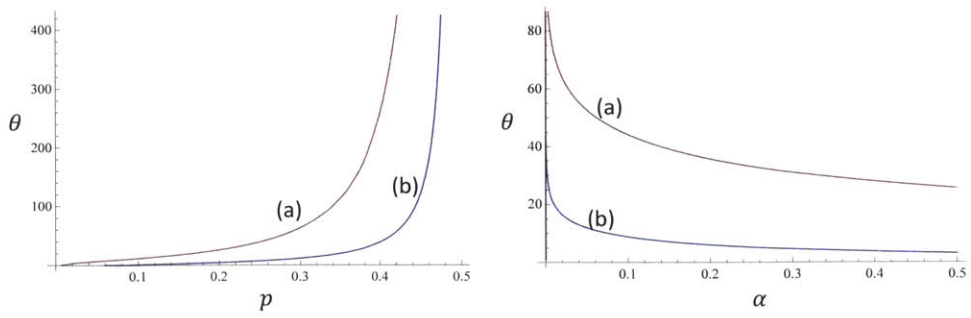

Figure 6.5. MM1 —Convergence rate analysis

■ 6.2 PageRank

In analyzing the web graph, PageRank is a frequently used measure to compute the importance of webpages. We are given a scalar parameter β and an underlying directed graph over n nodes with edge transitions given by matrix P. Let $\{X_t\}$ denote the random walk such that $\mathbb{P}(X_{t+1} = s | X_t = r) = \frac{\beta}{n} + (1-\beta)P_{rs}$. Thus, in every step, there is an β probability of jumping uniformly randomly to any other node in the network. The transition probability matrix of this pagerank random walk is given by

$$\frac{\beta}{n}\mathbf{1} \cdot \mathbf{1}^T + (1-\beta)P.$$

Algorithm results. In Figures 6.6 and 6.7, we show the results of applying our algorithm to estimate the PageRank of a random graph generated through the configuration model for $\beta = 0.15$. We choose parameters $\Delta = 0.02$, $\epsilon = 0.15$, and $\alpha = 0.2$. The graph on the left of Figure 6.6 shows the results of running the MCRT Algorithm for every single node separately. In this plot, we see that the MCRT Algorithm indeed obtains close estimates for nodes such that $\pi_i > \Delta$, and for nodes such that $\pi_i \leq \Delta$, the algorithm successfully categorizes the node as unimportant (i.e. $\hat{\pi}_i \leq \Delta$). In addition, we verify that the bias correction factor performs extremely well. We can compute the

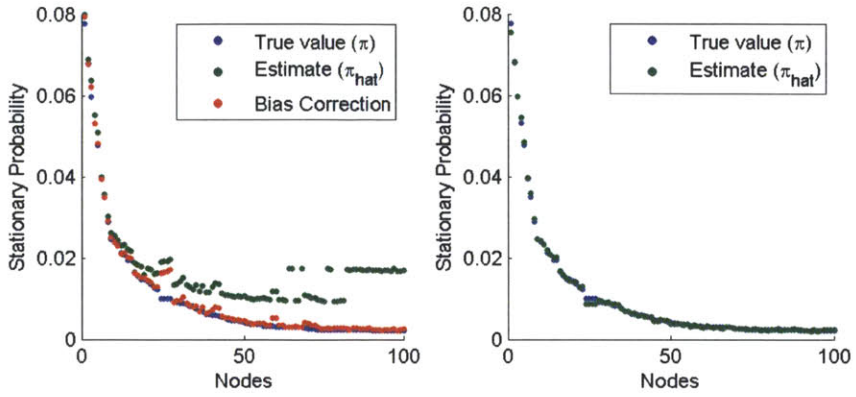

Figure 6.6. PageRank —(left) Estimates from the MCRT algorithm, (right) Estimates from the MCRT-Multi algorithm.

fundamental matrix Z for this example graph, and we verify that $\max_k(Z_{ii} - Z_{ki})$ is close to 1 for most nodes. By comparing with Theorem 4.2.1, we can verify that after the bias correction, the error decreases from approximately $\frac{\mathbb{P}_i(T_i > \theta^{(t)})}{\mathbb{E}_i[\hat{T}_i^{(t)}]}$ to near zero. For graphs such that $\max_k(Z_{ii} - Z_{ki})$ is large, we would expect that the bias correction will not be significant. The graph on the right of Figure 6.6 shows the results of running the MCRT-Multi Algorithm for i chosen to be the largest degree node. Figure 6.7 shows the final values for N, θ, and $\hat{p}$ for the execution of algorithm MCRT on each node. For all of these graphs, the nodes are sorted along the x-axis in decreasing order of π_i.

Algorithm dependence on Δ. Figure 6.8 shows the results of our algorithm as a function of Δ. The plot shows the results from three different nodes, chosen to illustrate the behavior on nodes with varying π_i. The graph on the left plots the estimates for the stationary probability and the graph on the right plots the total number of steps taken by the algorithm. The figures are shown on a log-log scale. These results support the same conclusions that were observed in the queueing setting. We observe that the computation time of the algorithm behaves as $\frac{1}{\Delta}$. The blue node has the largest stationary probability, therefore the algorithm will terminate at stopping condition (b) for small $\hat{p}$. Thus, as Δ decreases, the stationary probability estimate remains

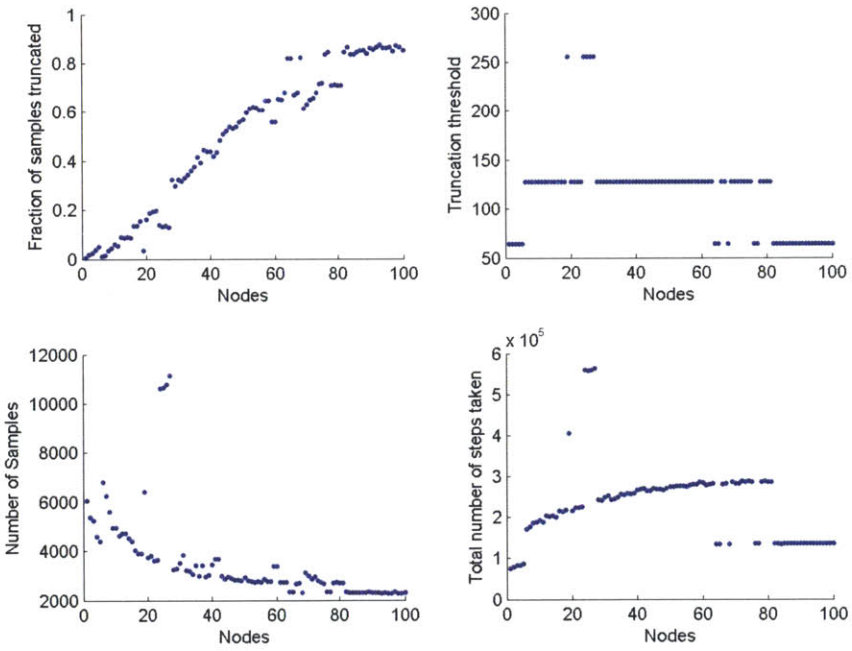

Figure 6.7. PageRank —Statistics from the MCRT algorithm.

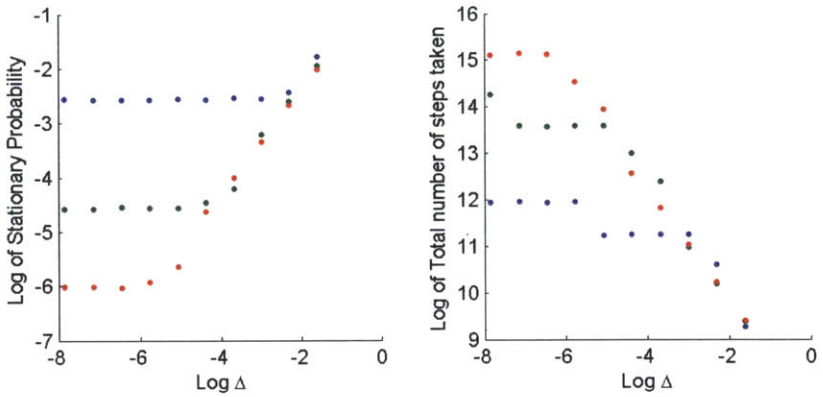

Figure 6.8. PageRank —Results of the MCRT algorithm as a function of Δ

the same, and the computation time increases only slowly in $\frac{1}{\Delta}$. The red node has the smallest stationary probability, $\pi_i \ll \Delta$. Therefore, the algorithm terminates at stopping condition (a), which is verified by observing that the estimate $\hat{\pi}_i$ behaves as Δ. For similar reasons, the computation behaves as $\frac{1}{\Delta}$. The green node shows both behaviors. When $\Delta > \pi_i$, then $\hat{\pi}_i$ behaves as Δ, and the computation time behaves as $\frac{1}{\Delta}$. When $\Delta < \pi_i$, then $\hat{\pi}_i$ concentrates around π_i and no longer decreases with Δ. The computation time levels off and grows slowly.

Algorithm dependence on β —Understanding properties of PageRank. We observe that the PageRank random walk is effectively a convex combination between a directed graph given by P, and a complete graph. We can thus analyze the tradeoff as a function of β, as the random walk becomes more or less similar to the complete graph. Much of the previous work on PageRank[4, 5, 7, 14, 20] only utilizes the property of the complete graph component of PageRank. Therefore, when β is large, then the algorithm performs very efficiently. However, when β is small, the algorithm scales with $\frac{1}{\beta}$. We first provide an analytic analysis and then show simluation results.

Lemma 6.2.1. *Given a markov chain $\{Y_t\}$ with transition probability matrix P, and a*

markov chain $\{X_t\}$ *with transition probability matrix*

$$\frac{\beta}{n}\mathbf{1}\cdot\mathbf{1}^T + (1-\beta)P,$$

the expected return time to a node i *can be expressed explicitly in terms of the return time to node* i *in the underlying directed graph, denoted by* T_i^Y*, and the weight of the complete graph component given by* β*.*

$$\begin{aligned}\mathbb{E}_i[T_i] &= \frac{1-\mathbb{E}_i[(1-\beta)^{T_i^Y}]}{\beta\mathbb{E}_U[(1-\beta)^{T_i^Y}]} \\ &= \frac{n(1-\mathbb{E}_i[(1-\beta)^{T_i^Y}])}{\beta\left(1+\sum_{j\in\Sigma\setminus\{i\}}\mathbb{E}_j[(1-\beta)^{T_i^Y}]\right)}.\end{aligned}$$

Proof. For the analysis, we "cut" the sequence $\{X_t\}$ based on when the β jumps are taken. Let $S_0 = 0$, and let S_k be the time that the k-th β jump is taken. Therefore,

$$(S_{k+1} - S_k) \sim \text{Geometric}(\beta)$$

For all $k \geq 1$, X_{S_k} is distributed uniformly over all nodes in the network, such that

$$\mathbb{P}(X_{S_k} = j) = \frac{1}{n}, \quad \text{for all} j.$$

Thus, we split up the sequence $\{X_t\}$ by $\{S_k\}$. Notice now that conditioned on the sequence $\{S_k\}$, the subsequences $\{X_{S_k}, X_{S_k+1}, \ldots X_{S_{k+1}-1}\}$ are independent. And in fact, they are also identically distributed for $k \geq 1$, because they all start uniformly distributed over every state, and evolve for a geometric number of steps according to the transition probability matrix P. Thus we want to understand the probability distribution over these "pieces" or geometric-length subsequences of the random walk.

We can generate each "piece" independently using the following process. Let $\{Y_l\}$

be a random walk on the underlying graph, i.e. using transition probability matrix P directly. Let the initial state Y_0 be distributed uniformly random over all nodes. Generate a random variable $G \sim$ Geometric(β), and take the first G steps in $\{Y_t\}$ as a realization of the subsequence $\{X_{S_k}, X_{S_k+1}, \ldots X_{S_{k+1}-1}\}$. We can "stitch" together these pieces to form a realization of $\{X_t\}$. We can similarly generate $\{X_0, X_1, \ldots X_{S_1-1}\}$ by simply setting the initial state $X_0 = Y_0 = i$, and taking a geometrically distributed number of steps on the Markov chain $\{Y_t\}$.

Let

$$T_i^Y = \min\{t \geq 1 : Y_t = i\},$$

$$T_i^0 = \min\{t \geq 0 : Y_t = i\},$$

and

$$Q = \min\{k \geq 0 : T_i < S_{k+1}\}.$$

$T_i^Y = T_i^0$ when $Y_0 \neq i$. Then we can represent T_i in the Markov chain $\{X_t\}$ as a sum of Q geometric random variables, and T_i^Y for $Y_0 \sim U$. Let G_k denote geometric random variables with parameter β. For $Q > 0$,

$$T_i = \sum_{k=0}^{Q-1} G_k + T_i^Y, \qquad \text{where } T_i^Y \text{ is distributed such that } Y_0 \sim U.$$

For $Q = 0$, $T_i = T_i^Y$ for $Y_0 = i$. Therefore,

$$\begin{aligned}
&\mathbb{E}_i[T_i] \\
&= \mathbb{P}(Q=0)\mathbb{E}_i[T_i^Y | T_i^Y < G_0] + \sum_{q=1}^{\infty} \mathbb{P}(Q=q)\mathbb{E}_U\left[\sum_{k=0}^{q-1} G_k + T_i^0 | Q=q\right] \\
&= \mathbb{P}(Q=0)\mathbb{E}_i[T_i^Y | T_i^Y < G_0] \\
&+ \sum_{q=1}^{\infty} \mathbb{P}(Q=q)\left(\mathbb{E}_i[G_k | G_k \le T_i^Y] + \sum_{k=1}^{q-1} \mathbb{E}_U[G_k | G_k \le T_i^0] + \mathbb{E}_U[T_i^Y | T_i^0 < G_q]\right)
\end{aligned}$$

Let G be a geometric random variable with parameter β, and let A be a discrete nonnegative random variable such that A is independent from G. By simply plugging in the expression for the distribution of G, we can show the following equalities:

$$\mathbb{P}(A < G_k) = \sum_{t=0}^{\infty} \mathbb{P}(A=t)\mathbb{P}(G_k > t) = \mathbb{E}[(1-\beta)^A].$$

$$\mathbb{E}[A | A < G] = \frac{\sum_{t=0}^{\infty} t\mathbb{P}(A=t)\mathbb{P}(G>t)}{\mathbb{P}(A<G)} = \frac{\mathbb{E}[A(1-\beta)^A]}{\mathbb{E}[(1-\beta)^A]}.$$

$$\begin{aligned}
\mathbb{E}[G | G \le A] &= \frac{\sum_{t=1}^{\infty}\sum_{g=1}^{t} g\mathbb{P}(A=t)\mathbb{P}(G=g)}{\mathbb{P}(A \ge G)} \\
&= \frac{\sum_{t=1}^{\infty}\mathbb{P}(A=t)\sum_{g=1}^{t} g(1-\beta)^{g-1}\beta}{\mathbb{P}(A \ge G)} \\
&= \frac{\sum_{t=1}^{\infty}\mathbb{P}(A=t)(\frac{1}{\beta}(1-(1-\beta)^t) - t(1-\beta)^t)}{\mathbb{P}(A \ge G)} \\
&= \frac{\frac{1}{\beta}(1-\mathbb{E}[(1-\beta)^A]) - \mathbb{E}[A(1-\beta)^A]}{1-\mathbb{E}[(1-\beta)^A]} \\
&= \frac{1}{\beta} - \frac{\mathbb{E}[A(1-\beta)^A]}{1-\mathbb{E}[(1-\beta)^A]}.
\end{aligned}$$

This governs the distribution of Q, such that

$$\mathbb{P}(Q=q) = (1-\mathbb{E}_i[(1-\beta)^{T_i^Y}]) \left(\prod_{k=1}^{q-1} (1-\mathbb{E}_U[(1-\beta)^{T_i^0}]) \right) \mathbb{E}_U[(1-\beta)^{T_i^0}].$$

where $\mathbb{P}(Q=0) = \mathbb{E}_i[(1-\beta)^{T_i^Y}]$ and $\mathbb{P}(Q=1) = (1-\mathbb{E}_i[(1-\beta)^{T_i^Y}])\mathbb{E}_U[(1-\beta)^{T_i^0}]$.

By substituting these expressions and doing algebraic manipulations, we can show that

$$\begin{aligned} \mathbb{E}_i[T_i] &= \frac{1-\mathbb{E}_i[(1-\beta)^{T_i^Y}]}{\beta \mathbb{E}_U[(1-\beta)^{T_i^Y}]} \\ &= \frac{n(1-\mathbb{E}_i[(1-\beta)^{T_i^Y}])}{\beta \left(1+\sum_{j\in\Sigma\setminus\{i\}} \mathbb{E}_j[(1-\beta)^{T_i^Y}]\right)} \end{aligned}$$

■

As expected, as

$$\beta \to 1, \quad \mathbb{E}_i[T_i] \to \frac{n}{\beta},$$

and as

$$\beta \to 0, \quad \mathbb{E}_i[T_i] \to \mathbb{E}_i[T_i^Y].$$

Therefore, we can see that β is a tuning parameter that controls whether the PageRank Markov chain is closer to the underlying directed graph, or to the complete graph. This bound depends on the exponential moments of T_i^Y. If $\mathbb{E}_i[T_i] \ll \frac{1}{\beta}$, then each iteration of our algorithm is on average shorter than a sample path in the standard local Pagerank algorithm (see Section 2.4).

Figures 6.9 and 6.10 show the results for computing PageRank where the underlying graph is a random graph generated using the configuration model for a power law degree distribution. As β increases, the graph approaches a complete graph. As β decreases, the graph approaches the underlying graph, i.e. the random graph generated by the

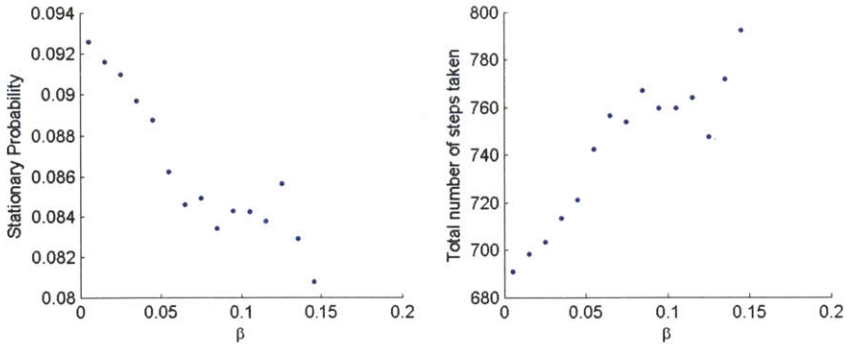

Figure 6.9. PageRank —Results of the MCRT algorithm as a function of β for a high degree node

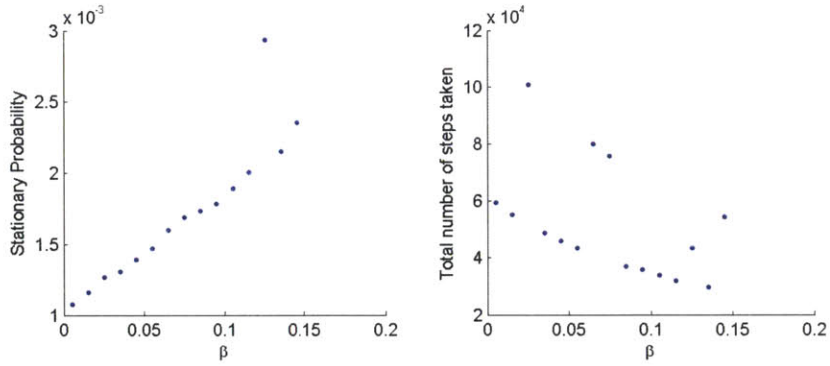

Figure 6.10. PageRank —Results of the MCRT algorithm as a function of β for a low degree node

configuration model.

Figure 6.9 shows the results for a node such that $\mathbb{E}[T_i^Y]$ is small (a large degree node). As β increases, the mass is shifted away and distributed to other nodes in the network, so the stationary probability decreases and the computation time increases. Figure 6.10 shows the results for a nodes such that $\mathbb{E}[T_i^Y]$ is large (a low degree node). As β increases, the mass from heavy weight nodes is distributed to this node, so the stationary probability increases and the computation time decreases. For all of these simulations, the parameters of the algorithm are chosen to be $\Delta = 0.02$, $\epsilon = 0.15$, and $\alpha = 0.2$.

■ 6.3 Magnet Graph

In this final example, we show a setting that illustrates the limitations of a local algorithm. This is an example that many existing algorithms would also perform poorly on because it mixes slowly. The Markov chain can be described as a walk over a finite section of the integers such that there are two attracting states, labeled in Figure 6.11 as states A and C. For all states left of state B, the random walk will drift towards state A with probablity $1 - \epsilon_1$, and for all states right of state B, the random walk will drift towards state C with probability $1 - \epsilon_2$. Due to this bipolar attraction, a random walk that begins on the left will tend to stay on the left, and similarly, a random walk that begins on the right will tend to stay on the right.

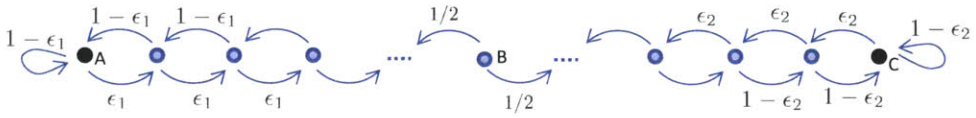

Figure 6.11. Magnet Markov chain.

Algorithm results. In Figures 6.12 and 6.13, we show the results of applying our algorithm to estimate the stationary probability of this Markov chain with $\epsilon_1 = \epsilon_2 = 0.3$. We chose a state space of 50 states such that the graph is symmetric: A = 1, B = 25, and C = 50. We use parameters $\Delta = 0.02$, $\epsilon = 0.15$, and $\alpha = 0.2$. The graph on the left of Figure 6.12 shows the results of running the MCRT Algorithm for every single node separately. As expected, the algorithm overestimates the stationary probabilities by almost two times the true value. This is due to the fact that the random samples have close to zero probability of sampling from the opposite half of the graph. Therefore the estimates are computed as if the opposite half of the graph did not exist. The bias correction does very little to improve the estimate, as the real problem is in the poor mixing properties of the graph. We compute the fundamental matrix Z for this Markov

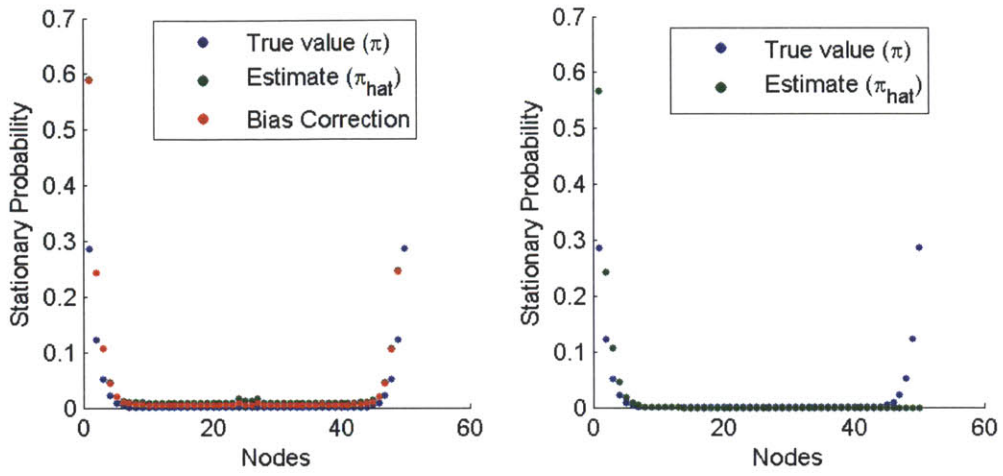

Figure 6.12. Magnet —(left) plots the results of the MCRT algorithm, (right) plots the results of the MCRT-Multi algorithm.

chain, and find that most of the absolute values of Z_{ij} are on the order of 10^9. This is consistent with our theoretical error bounds, which show that when $\max_j Z_{ij}$ is large, the estimates will be loose.

The graph on the right of Figure 6.12 shows the results of running the MCRT-Multi Algorithm for i chosen to be the leftmost node (i.e. node A = 1). As expected, the algorithm overestimates the leftmost nodes by a factor of two, and completely does not detect the nodes on the right. The severity of this effect will depend on the choice of ϵ_1 and ϵ_2. Similar to the example shown in Chapter 3, this is a graph on which any local algorithm will perform poorly on, since the algorithm will not be aware of the second attracting node on the opposite end. In fact, even the standard methods such as power iteration or MCMC will perform poorly on this graph, as it would take an incredibly large amount of time for the random walk to fully mix across the middle border. Figure 6.13 shows the final values for N, θ, and $\hat{p}$ for the execution of algorithm MCRT on each node.

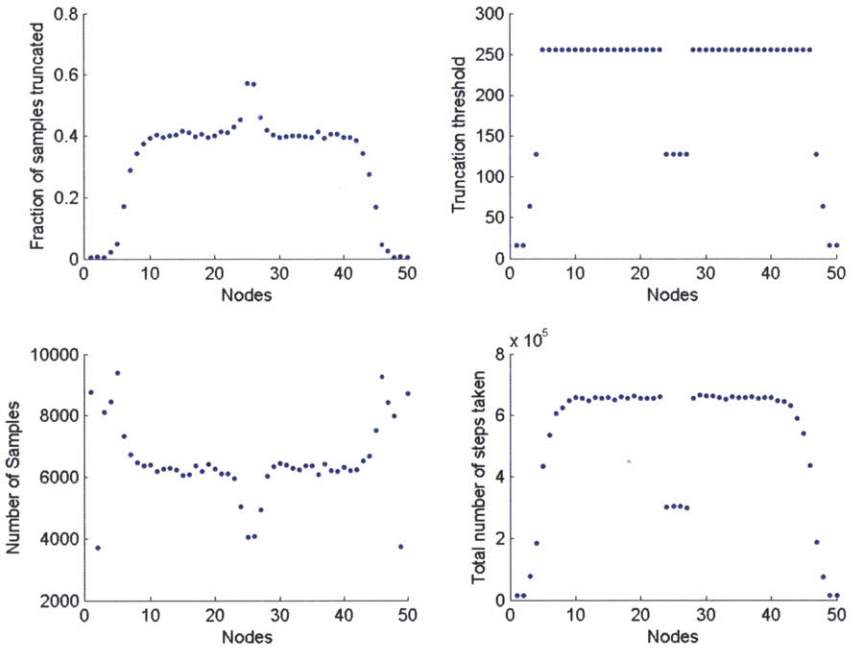

Figure 6.13. Magnet —Statistics from the MCRT algorithm.

Chapter 7

Conclusion

This thesis presents a local algorithm for estimating the stationary probability of a node in a Markov chain. The algorithm is a *truncated Monte Carlo* method, sampling return paths to the node of interest. The key insight it utilizes is that the stationary probability of a node is inversely proportional to the expected return time to the node $\mathbb{E}_i[T_i]$. We provide theoretical guarantees for the tightness of approximation and convergence time of our algorithm. The estimate we obtain is an upper bound with high probability, and for Markov chains that mix well, we further show that the estimate will be tight. Given a threshold Δ, our algorithm obtains close estimates for high probability nodes, and obtains reasonable upper bounds on low probability nodes. The analysis and guarantees rely on the property that for a large set of positive recurrent countable state space Markov chains, the distribution over return times to a node concentrates around the mean. We showed that this concentration rate is related to the mixing properties of the Markov chain.

The algorithm has many practical benefits. First, it can be implemented easily in a distributed and parallelized fashion, as it only involves sampling random walks using neighbor queries. Second, it only requires a neighborhood around the node of interest, scaling with $O\left(\frac{1}{\Delta}\right)$. Third, it only requires computation on behalf of the nodes of interest. The computation itself only involves counting and taking an average, thus it is simple and memory efficent. We showed an extension of our algorithm to reuse the

computation to estimate the probablities of multiple nodes simultaneously. It requires computation to be performed at those additional nodes of interest. We show through simulation that it performs well when the graph has reasonable mixing properties.

There is considerable room for the further development of local algorithms for computing global properties of graphs, and for discovering or searching for nodes that satisfy certain properties. These techniques will become important as the scale of the networks we operate over is growing at a fast rate. This thesis provides a step in this direction towards understanding which global properties can be computed in a local manner.

Bibliography

[1] D. Aldous and J. Fill. Reversible Markov chains and random walks on graphs: Chapter 2 (General Markov chains). book in preparation. *URL for draft at http://www.stat.berkeley.edu/ aldous/RWG/Chap2.pdf*, pages 7, 19–20, 1999.

[2] Reid Andersen, Christian Borgs, Jennifer Chayes, John Hopcraft, Vahab S. Mirrokni, and Shang-Hua Teng. Local computation of pagerank contributions. In *Proceedings of the 5th international conference on Algorithms and models for the web-graph*, WAW'07, pages 150–165, Berlin, Heidelberg, 2007. Springer-Verlag. ISBN 3-540-77003-8, 978-3-540-77003-9. URL `http://dl.acm.org/citation.cfm?id=1777879.1777891`.

[3] S. Assmussen and P. Glynn. *Stochastic Simulation: Algorithms and Analysis (Stochastic Modelling and Applied Probability)*. Springer, 2010.

[4] K. Avrachenkov, N. Litvak, D. Nemirovsky, and N. Osipova. Monte Carlo methods in PageRank computation: When one iteration is sufficient. *SIAM Journal on Numerical Analysis*, 45(2):890–904, 2007. doi: 10.1137/050643799. URL `http://epubs.siam.org/doi/abs/10.1137/050643799`.

[5] Bahman Bahmani, Abdur Chowdhury, and Ashish Goel. Fast incremental and personalized PageRank. *Proc. VLDB Endow.*, 4(3):173–184, December 2010. ISSN 2150-8097. URL `http://dl.acm.org/citation.cfm?id=1929861.1929864`.

[6] Dimitris Bertsimas, David Gamarnik, and John N. Tsitsiklis. Geometric bounds for stationary distributions of infinite Markov chains via Lyapunov functions. Technical report, MIT Sloan School of Management, 1998.

[7] Christian Borgs, Michael Brautbar, Jennifer T. Chayes, and Shang-Hua Teng. A sublinear time algorithm for PageRank computations. In *WAW*, pages 41–53, 2012.

[8] Michael Brautbar and Michael J Kearns. Local algorithms for finding interesting individuals in large networks. *Innovations in Computer Science*, 2010.

[9] Ozan Candogan, Kostas Bimpikis, and Asuman Ozdaglar. Optimal pricing in networks with externalities. *Operations Research*, 60(4):883 905, 2012. URL `http://or.journal.informs.org/content/60/4/883.abstract`.

[10] Georgios C. Chasparis and Jeff S. Shamma. Control of preferences in social networks. In *CDC*, pages 6651–6656, 2010.

[11] Yen-Yu Chen, Qingqing Gan, and Torsten Suel. Local methods for estimating pagerank values. In *Proceedings of the thirteenth ACM international conference on Information and knowledge management*, pages 381–389. ACM, 2004.

[12] Persi Diaconis. The markov chain monte carlo revolution. *Bulletin of the American Mathematical Society*, 46(2):179–205, 2009.

[13] Persi Diaconis and Laurent Saloff-Coste. What do we know about the metropolis algorithm? *Journal of Computer and System Sciences*, 57(1):20–36, 1998.

[14] Daniel Fogaras, Balazs Racz, Karoly Csalogany, and Tamas Sarlos. Towards scaling fully personalized PageRank: Algorithms, lower bounds, and experiments. *Internet Mathematics*, 2(3):333–358, 2005.

[15] FG Foster. On the stochastic matrices associated with certain queuing processes. *The Annals of Mathematical Statistics*, 24(3):355–360, 1953.

[16] G.H. Golub and C.F. Van Loan. *Matrix Computations*. Johns Hopkins Studies in the Mathematical Sciences. Johns Hopkins University Press, 1996. ISBN 9780801854149.

[17] B. Hajek. Hitting-time and occupation-time bounds implied by drift analysis with applications. *Advances in Applied probability*, pages 502–525, 1982.

[18] W Keith Hastings. Monte carlo sampling methods using markov chains and their applications. *Biometrika*, 57(1):97–109, 1970.

[19] Taher H. Haveliwala. Topic-sensitive pagerank: A context-sensitive ranking algorithm for web search. Technical Report 2003-29, Stanford InfoLab, 2003. URL `http://ilpubs.stanford.edu:8090/750/`. Extended version of the WWW2002 paper on Topic-Sensitive PageRank.

[20] Glen Jeh and Jennifer Widom. Scaling personalized web search. In *Proceedings of the 12th international conference on World Wide Web*, WWW '03, pages 271–279, New York, NY, USA, 2003. ACM. ISBN 1-58113-680-3. doi: 10.1145/775152.775191. URL `http://doi.acm.org/10.1145/775152.775191`.

[21] Sepandar Kamvar, Taher Haveliwala, Christopher Manning, and Gene Golub. Exploiting the block structure of the web for computing pagerank. *Stanford University Technical Report*, 2003.

[22] JR Koury, DF McAllister, and William J Stewart. Iterative methods for computing stationary distributions of nearly completely decomposable markov chains. *SIAM Journal on Algebraic Discrete Methods*, 5(2):164–186, 1984.

[23] David Asher Levin, Yuval Peres, and Elizabeth Lee Wilmer. *Markov chains and mixing times.* Amer Mathematical Society, 2009.

[24] Semkow Thomas M., Pomm Stefaan, Jerome Simon, and Strom Daniel J., editors. *Applied Modeling and Computations in Nuclear Science.* American Chemical Society, Washington, DC, 2006. doi: 10.1021/bk-2007-0945. URL `http://pubs.acs.org/doi/abs/10.1021/bk-2007-0945`.

[25] N. Metropolis, A.W. Rosenbluth, M.N. Rosenbluth, A.H. Teller, and E. Teller. Equation of state calculations by fast computing machines. *The journal of chemical physics*, 21:1087, 1953.

[26] SP Meyn and RL Tweedie. Markov chains and stochastic stability, 1993.

[27] Sahand Negahban, Sewoong Oh, and Devavrat Shah. Iterative ranking from pairwise comparisons. *CoRR*, abs/1209.1688, 2012.

[28] M. E. J. Newman. *Networks: an introduction.* Oxford University Press, Oxford; New York, 2010. ISBN 9780199206650 0199206651.

[29] Lawrence Page, Sergey Brin, Rajeev Motwani, and Terry Winograd. The PageRank citation ranking: Bringing order to the web. Technical Report 1999-66, November 1999.

[30] Atish Das Sarma, Sreenivas Gollapudi, and Rina Panigrahy. Estimating pagerank on graph streams. *J. ACM*, 58(3):13, 2011.

[31] Atish Das Sarma, Anisur Rahaman Molla, Gopal Pandurangan, and Eli Upfal. Fast distributed pagerank computation. *CoRR*, abs/1208.3071, 2012.

[32] D. Shah and T. Zaman. Rumors in a network: Who's the culprit? *Information Theory, IEEE Transactions on*, 57(8):5163 –5181, aug. 2011.

[33] William J Stewart. *Introduction to the numerical solution of Markov chains*, volume 41. Princeton University Press Princeton, 1994.